


普通高等教育"十一五"国家级规划教材

国家精品课程配套教材

可下载教学资料

http://www.tup.tsinghua.edu.cn

高等学校教材

计算机应用

# Web技术导论

## （第2版）

郝兴伟 编著

清华大学出版社



普通高等教育“十一五”国家级规划教材
高等学校教材·计算机应用

# Web 技术导论

## （第 2 版）

郝兴伟　编著

清华大学出版社
北　京

## 内 容 简 介

全书共分为6章，分别介绍了Web技术的相关概念、核心技术及Web的最新进展；Web服务器的架设和管理；HTML和XML标记语言，XML相关技术及开发环境；网页设计与制作；Web客户端开发技术，包括JavaScript和AJAX技术；Web服务器开发，包括Java技术、JSP技术等内容。

本书内容广泛、新颖，逻辑严谨，具有很强的专业性、技术性和可操作性。本书是为高等学校计算机应用、信息管理及电子商务等专业的学生编写的Web技术、开发及应用的综合性书籍，也可以作为非计算机专业高年级学生学习计算机技术的教材，同时还可作为科技人员和IT从业人员了解Web的较好的综合性参考书。



**图书在版编目(CIP)数据**

Web技术导论/郝兴伟编著.—2版.—北京：清华大学出版社，2009.4
(高等学校教材·计算机应用)
ISBN 978-7-302-19371-5

Ⅰ.W…　Ⅱ.郝…　Ⅲ.主页制作－程序设计－高等学校－教材　Ⅳ.TP393.092

中国版本图书馆CIP数据核字(2009)第011522号

**责任编辑**：付弘宇　王冰飞
**责任校对**：梁　毅
**责任印制**：

**出版发行**：清华大学出版社　　**地　　址**：北京清华大学学研大厦A座
http://www.tup.com.cn　　**邮　　编**：100084
**社　总　机**：010-62770175　　**邮　　购**：010-62786544
**投稿与读者服务**：010-62776969，c-service@tup.tsinghua.edu.cn
**质　量　反　馈**：010-62772015，zhiliang@tup.tsinghua.edu.cn
**印　刷　者**：
**装　订　者**：
**经　　销**：全国新华书店
**开　　本**：185×260　**印　张**：24.5　　**字　　数**：612千字
**版　　次**：2009年3月第2版　　**印　　次**：2009年3月第1次印刷
**印　　数**：1～　000
**定　　价**：　.00元

本书如存在文字不清、漏印、缺页、倒页、脱页等印装质量问题，请与清华大学出版社出版部联系调换。
联系电话：010-62770177转3103　　产品编号：

高等学校教材·计算机应用

# 编审委员会成员

（按地区排序）

| | | |
|---|---|---|
| 南京邮电学院 | 朱秀昌 | 教授 |
| 苏州大学 | 龚声蓉 | 教授 |
| 江苏大学 | 宋余庆 | 教授 |
| 武汉大学 | 何炎祥 | 教授 |
| 华中科技大学 | 刘乐善 | 教授 |
| 中南财经政法大学 | 刘腾红 | 教授 |
| 华中师范大学 | 王林平 | 副教授 |
| | 魏开平 | 副教授 |
| | 叶俊民 | 教授 |
| 国防科技大学 | 赵克佳 | 教授 |
| | 肖　侬 | 副教授 |
| 中南大学 | 陈松乔 | 教授 |
| | 刘卫国 | 教授 |
| 湖南大学 | 林亚平 | 教授 |
| | 邹北骥 | 教授 |
| 西安交通大学 | 沈钧毅 | 教授 |
| | 齐　勇 | 教授 |
| 长安大学 | 巨永峰 | 教授 |
| 西安石油学院 | 方　明 | 教授 |
| 西安邮电学院 | 陈莉君 | 教授 |
| 哈尔滨工业大学 | 郭茂祖 | 教授 |
| 吉林大学 | 徐一平 | 教授 |
| | 毕　强 | 教授 |
| 长春工程学院 | 沙胜贤 | 教授 |
| 山东大学 | 孟祥旭 | 教授 |
| | 郝兴伟 | 教授 |
| 山东科技大学 | 郑永果 | 教授 |
| 中山大学 | 潘小轰 | 教授 |
| 厦门大学 | 冯少荣 | 教授 |
| 福州大学 | 林世平 | 副教授 |
| 云南大学 | 刘惟一 | 教授 |
| 重庆邮电学院 | 王国胤 | 教授 |
| 西南交通大学 | 杨　燕 | 副教授 |

# 出版说明

改革开放以来，特别是党的十五大以来，我国教育事业取得了举世瞩目的辉煌成就，高等教育实现了历史性的跨越，已由精英教育阶段进入国际公认的大众化教育阶段。在质量不断提高的基础上，高等教育规模取得如此快速的发展，创造了世界教育发展史上的奇迹。当前，教育工作既面临着千载难逢的良好机遇，同时也面临着前所未有的严峻挑战。社会不断增长的高等教育需求同教育供给特别是优质教育供给不足的矛盾，是现阶段教育发展面临的基本矛盾。

教育部一直十分重视高等教育质量工作。2001 年 8 月，教育部下发了《关于加强高等学校本科教学工作，提高教学质量的若干意见》，提出了十二条加强本科教学工作提高教学质量的措施和意见。2003 年 6 月和 2004 年 2 月，教育部分别下发了《关于启动高等学校教学质量与教学改革工程精品课程建设工作的通知》和《教育部实施精品课程建设提高高校教学质量和人才培养质量》文件，指出“高等学校教学质量和教学改革工程”是教育部正在制定的《2003—2007 年教育振兴行动计划》的重要组成部分，精品课程建设是“质量工程”的重要内容之一。教育部计划用五年时间（2003—2007 年）建设 1500 门国家级精品课程，利用现代化的教育信息技术手段将精品课程的相关内容上网并免费开放，以实现优质教学资源共享，提高高等学校教学质量和人才培养质量。

为了深入贯彻落实教育部《关于加强高等学校本科教学工作，提高教学质量的若干意见》精神，紧密配合教育部已经启动的“高等学校教学质量与教学改革工程精品课程建设工作”，在有关专家、教授的倡议和有关部门的大力支持下，我们组织并成立了“清华大学出版社教材编审委员会”（以下简称“编委会”），旨在配合教育部制定精品课程教材的出版规划，讨论并实施精品课程教材的编写与出版工作。“编委会”成员皆来自全国各类高等学校教学与科研第一线的骨干教师，其中许多教师为各校相关院、系主管教学的院长或系主任。

按照教育部的要求，“编委会”一致认为，精品课程的建设工作从开始就要坚持高标准、严要求，处于一个比较高的起点上；精品课程教材应该能够反映各高校教学改革与课程建设的需要，要有特色风格、有创新性（新体系、新内容、新手段、新思路，教材的内容体系有较高的科学创新、技术创新和理念创新的含量）、先进性（对原有的学科体系有实质性的改革和发展、顺应并符合新世纪教学发展的规律、代表并引领课程发展的趋势和方向）、示范性（教材所体现的课程体系具有较广泛的辐射性和示范性）和一定的前瞻

性。教材由个人申报或各校推荐(通过所在高校的“编委会”成员推荐),经“编委会”认真评审,最后由清华大学出版社审定出版。

目前,针对计算机类和电子信息类相关专业成立了两个“编委会”,即“清华大学出版社计算机教材编审委员会”和“清华大学出版社电子信息教材编审委员会”。首批推出的特色精品教材包括:

(1) 高等学校教材·计算机应用——高等学校各类专业,特别是非计算机专业的计算机应用类教材。

(2) 高等学校教材·计算机科学与技术——高等学校计算机相关专业的教材。

(3) 高等学校教材·电子信息——高等学校电子信息相关专业的教材。

(4) 高等学校教材·软件工程——高等学校软件工程相关专业的教材。

(5) 高等学校教材·信息管理与信息系统。

(6) 高等学校教材·财经管理与计算机应用。

清华大学出版社经过 20 多年的努力,在教材尤其是计算机和电子信息类专业教材出版方面树立了权威品牌,为我国的高等教育事业做出了重要贡献。清华版教材形成了技术准确、内容严谨的独特风格,这种风格将延续并反映在特色精品教材的建设中。

**清华大学出版社教材编审委员会**
**E-mail:dingl@tup.tsinghua.edu.cn**

# 第2版前言

今天,Internet已经成为一种最基本的社会基础设施,它几乎渗透到了现代社会的每一个角落。无论是IT专业人员、其他工作人员还是一般计算机用户,互联网已经成为人们最主要的通信、获取信息和发布信息的媒体。互联网应用的普及推动了人们对学习和了解Internet相关技术的社会需求。但是,走进书店或在Internet上查询,关于互联网的书籍铺天盖地,令人眼花缭乱,以至于我们无所适从。为此,我想编写一本介绍互联网开发和应用的综合性书籍,使大家对目前的互联网,特别是Web技术从概念、原理和应用上有一个总体的了解和把握,这就是本书第1版写作的初衷。

从2005年本书第1版的出版到现在,三年过去了,《Web技术导论》一书受到了许多老师的认可,被选作他们的教科书。我也非常高兴地收到了多位任课教师的邮件,与我交流书中的相关技术,有些老师还非常诚恳地对本书提出了一些良好建议,例如,增加有关Web服务、SOA等最新Web概念的内容,去掉操作性的多媒体制作章节,等等。这些良好的建议和这几年来我在Web开发中的一些新的认识和体会促使自己决定对第1版的内容进行彻底的修订,增加更多新技术的讲解,特别是Web环境下的软件体系结构、设计模式、开发模式、AJAX技术等新的内容,从而使本书能够紧跟互联网的发展步伐。

本书作为一本导论性质的书籍,全面介绍互联网的发展历史、最新的科学进展、Web的工作原理、计算模式和软件体系结构的演变、Web核心技术、互联网语言、Web设计模式、Web客户端开发、Web服务端开发等内容。相信这样的内容安排对大多数读者都会有所帮助。如果你是一个初学者,这本书会为你答疑解惑;如果你是一个初级的开发人员,这本书可以为你建立一个Web开发的基本框架,引领你进入Web开发的广阔天地;如果你是一个高级开发人员,本书的综合性内容也会为你阅读其他专业知识做一个基本知识的铺垫。

本书与第1版一样,仍然分成6章,主要内容如下:

第1章 Web基础。介绍互联网的发展和相关概念,Web的工作机理以及Java技术、XML、Web服务等Web核心技术,还介绍了计算机软件体系架构的演变和SOA体系架构的思想,最后介绍了Web 2.0和语义Web的发展。

第2章 Web服务器的架设和管理。首先介绍了操作系统和Web服务器的概念,然后介绍了Windows平台下的Web服务器的架设和管理,主要讲解了Windows Server平台中的IIS,对IIS的讲解比较简单,易于理解。在理解了Web服务的管理后,重点讲解Apache Tomcat的架设和管理以及Web应用的部署等。Apache是开发

Web 应用最常用的运行平台。最后对 Web 服务器的远程管理进行了讲解。

第 3 章 HTML 和 XML 基础。首先介绍了标记语言的概念,介绍 HTML 标记语言的基本语法,并安排了大量实例来说明每种元素的含义和使用。对 HTML 和 XML 的本质区别进行了深入的分析和总结。讲解了 XML 相关的规范,包括可扩展样式语言 XSL、XML 路径语言 XPath、XML 查询语言 XQuery、可扩展连接语言 XLL、XML 文档对象模型 DOM 与简单应用程序接口 SAX,并对它们之间的关系进行了总结,这些内容对大家理解以 XML 为核心的 Web 技术具有重要意义。

第 4 章网页设计与制作。网页是 Web 应用的主要用户界面,在 HTML 和 XML 基础上,加强了网页设计的讲解,包括页面功能与内容设计、页面布局设计、页面视觉设计以及页面效果设计等。然后,介绍了可视化的网页制作工具 FrontPage。

第 5 章客户端开发。首先讲解了 Web 浏览器的基本工作原理,然后讲解了客户端脚本程序设计语言 JavaScript、浏览器对象模型 BOM、HTML 文档对象模型 DOM、Web 交互的内容。增加了 AJAX 技术的讲解,最后详细讲解了两个综合性客户端开发实例。

第 6 章服务端开发。首先介绍了 B/S 三层结构的概念,然后重点讲解了 Java 技术及其在 Web 开发中的应用,包括 Java 程序设计语言、Java Applet、JavaBeans、Servlet 服务器程序、JSP 技术以及 MVC 设计模式。在 JSP 技术中,讲解了 JSP 的语法、内置对象、数据库操作、图形操作等许多实用的内容。最后,讲解了在线聊天 Web 应用的整个开发过程,同时对常用的 Java 开发工具进行了介绍。

作为互联网的用户和 Web 技术的开发者和实践者,同时,作为一个公司派的高校教师,虽然我的初衷是要使本书既包含较广的理论知识,又有很好的技术内容,但是要真正地将理论和技术结合起来是很困难的。一方面是 Web 相关的技术实在太多,作者本人的知识面和认识有限,加之时间仓促;再者是考虑到读者的实际应用需求非常多样,很想把一些更实用的软件代码介绍给大家,并进行讲解,但是受到篇幅的限制,也不能如愿。

在本书写作的过程中,我要非常感谢我的同事巩裕伟教授,他是一名优秀的老师,总是将计算机技术深入浅出地传授给学生,受到学生的普遍欢迎;同时,他还是一位很好的程序员,编写了大量的 Java、JSP、VB 程序和数据库应用系统;另外,他还是一位出色的作者,我们合作编写过许多计算机的书籍。我要感谢我的同事焦文江老师,他对网络环境有着很深入的研究,对网络设备非常熟悉,对待工作总是认真负责。同时我要感谢参与本书编写工作的杨兴强、吕刚、阚铮和李蕴几位老师。还要感谢我的学生苏雪、常跃峰、崔旭和朱岩,编写了大量的程序代码,祝愿他们在以后的工作和生活中一切顺利,取得更大的成绩。还要感谢孟祥旭教授、王海洋教授、马军教授、张彩明教授、徐秋亮教授、龙世立研究员,作为领导、同事和朋友,他们在学术上和事业上都给了我很大的帮助。还要感谢山东大学研究生院的立项资助。最后,要感谢 Internet 本身,是它为我们提供了海量的信息和如此快速、便捷的交流平台。

由于本书涉及的内容非常广泛,在深度和广度上很难做到完美,加之作者水平有限,书中难免存在错误和不足,请读者批评指正。

作者 E-mail:hxw@sdu.edu.cn。

本书配套课件等资源可以从清华大学出版社网站 http://www.tup.tsinghua.edu.cn 下载。

作　者

2008 年 11 月

# 目录

# 第1章

# Web基础

今天,互联网已经成为使用最广泛的传播媒体,它正在改变着人们的工作、生活和娱乐方式。通过 Internet,人们可以收发电子邮件,进行网上学习、网上购物,开展网上讨论、网上聊天、网络游戏,观看网上电视等。通过 Internet,人们还可以在网上发布信息、检索信息、开展商务活动等。随着网络带宽的不断提高,网络基础设施的进一步完善,联网主机和互联网用户正在以惊人的速度增长,Internet 就像空气一样正在渗入到我们生活的每一个角落。本章简要介绍 Web 的有关概念、相关技术以及 Web 的发展趋势。

## 1.1 Internet 与万维网

1946 年,第一台电子计算机“爱尼亚克”(ENIAC)在美国宾夕法尼亚大学莫尔电子工程学院诞生。这种计算技术的革命,透出了数字信息时代的第一缕曙光。随后,微电子技术和计算机技术的发展日新月异。为了进一步提高计算机的使用效率,人们需要将不同的计算机连接起来,传递数据,共享资源,因此计算机网络诞生了。

20 世纪 60 年代,出现了各式各样的计算机网络,来实现计算机之间的通信和资源共享。1969 年,美国国防部高级研究计划署 ARPA 资助了一个有关广域网络的项目,开发一个称作阿帕网(ARPANet)的网络,它的主要思想是构建一个没有中央控制结点的计算机网络,以便使军事计算机系统在受到打击后不会因为部分毁坏而导致整个计算机网络的瘫痪。

1969 年 11 月 21 日中午,6 名科学家聚会美国加利福尼亚大学洛杉矶分校的计算机实验室,观看这里的一台计算机与远在千里之外的斯坦福研究所的另一台计算机联通。这是一个历史性的时刻,正像 20 年后《时代》周刊的评论:这些研究者根本没有想到,他们不只是连接了两台计算机,而是宣告了网络世界的到来。

到 1970 年,ARPANet 已初具雏形,已经将加利福尼亚州大学洛杉矶分校、加州大学圣巴巴拉分校、斯坦福大学、犹他州大学四所大学的 4 台计算机以分组交换协议连接起来,实现了不同型号、不同操作系统、不同数据格式、不同终端的计算机之间的通信和资源共享。

1972 年,ARPANet 已建成 40 多个网点,开发出了三项主要的功能,即以后被广泛使用的电子邮件、远程登录和文件传输。1974 年,著名的 TCP/IP 协议研究成功,彻底解决了不同计算机系统之间的通信问题,计算机互联的主要障碍被解决。

1975 年,ARPANet 的运行管理移交给美国国防通信局(DCA)。1982 年 DCA 将

ARPANet 各站点的通信协议全部转为 TCP/IP,同时 ARPANet 被分成两部分,一部分作为军用,称为 MILnet,另一部分作为民用,ARPANet 开始从一个实验型网络向实用型网络转变,从而成为全球 Internet 正式诞生的标志。

如果把 Internet 的发展划分阶段的话,那么 1969～1984 年的这个时期可以看成是 Internet 的提出、研究和试验阶段,这时的 Internet 以 ARPANet 为主干网。由于 ARPANet 采用离散结构,不设中央网络控制设备,实现了网络渠道的多样性,从而减少了系统彻底崩溃的可能性,网络的生存能力得到了保证,实现了 ARPA 的最初构想。

后来,Internet 的发展超出了任何人的想象。从 1984 年到 1992 年可以看做是 Internet 的实用发展阶段。为了使全美国的科学家和工程师都能够共享那些过去只有军事部门和少数科学家才能够使用的超级计算机设施,美国国家科学基金会 NSF(National Science Foundation)于 1985 年提供巨资建设了全美 5 个超级计算中心,同时建设了将这些超级计算中心和各科研机构相连的高速信息网络 NSFnet。1986 年 NSFnet 成功地成为 Internet 的第二个骨干网。NSFnet 对 Internet 的推广起到了巨大的推动作用,它使得 Internet 不再是仅有科学家、工程师、政府部门使用的网络,Internet 进入了以资源共享为中心的实用服务阶段。以连接 NSFnet 的局域网数量为例,1988 年 7 月只有 170 个,到 1992 年 1 月这一数量就发展到 4500 个。

1992 年以后 Internet 开始进入它的商业化发展阶段,Internet 用户开始向全世界扩展,并以每月 15%的速度迅速增长,每 30 分钟就有一个网络连入 Internet。随着网上通信量的急剧增长,Internet 开始不断采用新的技术以适应发展的需求,其主干网由政府部门资助开始向商业计算机公司、通信公司转化。

在 Internet 商业化的过程中,万维网(World Wide Web,WWW)的出现,使 Internet 的使用更简单、更方便,开创了 Internet 发展的新时期。1989 年,在瑞士日内瓦粒子物理研究实验室欧洲核子研究中心(CERN)工作的蒂姆·伯纳斯·李(Tim Berners-Lee)首先提出了 WWW 的概念,并且成功地开发出世界上第一个万维网服务器和第一个万维网客户机。同年底,蒂姆为他的发明正式定名为 World Wide Web(万维网);1991 年 5 月万维网在因特网上首次露面,立即引起轰动,迅速被广泛推广应用。

在 WWW 的发展中,还有一位杰出的人物,他就是马克·安德森,是他改造了 Internet 的使用界面。在早期,万维网只有文字,没有图像、声音,也没有色彩。对普通用户来说,仍缺乏一种简单的使用界面。安德森在就读伊利诺斯大学时,开始在学校里的国家超级计算中心(NCSA)兼职工作,由于感觉到 Internet 界面的难于使用,他和同事贝纳一起合作,经过 6 个星期的辛苦工作,终于在 1993 年 1 月有了初步成果,写出了 UNIX 版的马赛克(Mosaic)浏览器。

美国著名信息专家、《数字化生存(Being Digital)》一书的作者尼古拉·尼葛洛庞帝(Nicholas Negroponte)教授认为:1989 年是互联网历史上划时代的分水岭,这一年出现的万维网技术给 Internet 赋予了强大的生命力,把 Internet 带入了一个崭新的时代。

## 1.2 Web 概述

1990 年,瑞士日内瓦世界上最大的粒子物理研究实验室欧洲核子研究中心 CERN(the European Organization for Nuclear Research)提出了 World Wide Web(WWW)的概念,它

是 Internet 技术、超文本技术和多媒体技术相结合的产物。当时,核物理的研究是分散在不同国家进行的,各地的研究人员通过计算机网络和 Internet 进行学术交流。在 Internet 中信息交流还没有一种统一的手段,因此,根据交流的信息不同(如图片、文字等)需要调用不同的 Internet 服务,很不方便。1989 年 3 月,CERN 的 Tim Berners-Lee 开发了一个超级文本系统,1990 年底,第一个基于字符界面的 Web 客户浏览程序开发成功,1991 年 3 月,客户浏览程序开始在 Internet 上运行,1991 年底 CERN 向高能物理学界宣布了 Web 服务。

### 1.2.1 什么是 Web

什么是 Web 呢?从万维网诞生起,人们并没有给它一个确切的定义。我们可以从 Internet 的构成和服务来理解 Web。

Internet 是一个网络上的网络,或者说是一个全球范围的网间网。在 Internet 中,分布了成千上万的无以计数的计算机,这些计算机扮演的角色和所起的作用不同。有的计算机可以收发用户的电子邮件,有的可以为用户传输文件,有的负责对域名进行解析,更多的机器则用于组织并展示本网络的信息资源,方便用户的获取。所有这些承担服务任务的计算机我们统称为服务器。根据服务的特点来区分,又分为邮件服务器、文件传输服务器、DNS 服务器、Web 服务器等。

所谓 Web 服务器,就是将本地的信息用超级文本组织,向用户提供在 Internet 上进行信息浏览服务的计算机。因此,Web 或者说 World Wide Web,是由 Internet 中称为 Web 信息服务器的计算机组成的,它们由那些希望通过 Internet 发布信息的机构提供并管理。在 Web 世界里,每一个 Web 服务器除了提供自己独特的信息服务外,还可以用超链接指向其他的 Web 服务器,那些 Web 服务器又可以指向更多的 Web 服务器,这样一个全球范围的由 Web 服务器组成的 World Wide Web(万维网)就形成了。

### 1.2.2 Web 的工作原理

上面我们已经多次提到 Web 服务器和客户机,那么什么是 Web 服务器和 Web 客户机呢?我们知道,万维网是由分布在 Internet 中的 Web 服务器组成的。要使一台计算机成为一台 Web 服务器,一般需要服务器操作系统,例如 UNIX、Windows Server 2003、Linux 等网络操作系统,并且还要安装专门的信息服务器程序,如 Windows 中的 Internet 信息服务器 IIS(Internet Information Server)、Apache Tomcat 等。要成为 Web 客户机很简单,将计算机连接到 Internet,并安装通用的客户端软件,即 Internet Explorer、Netscape、Maxthon(遨游)、Firefox(火狐狸)等浏览器程序。

万维网的运行是一种典型的浏览器/服务器(Browser/Server,B/S)模式。典型的 B/S 体系结构将计算机应用分成三个层次,即客户端浏览器层、Web 服务器层和数据库服务器层。B/S 体系结构有许多优点,它简化了客户端的维护,所有的应用逻辑都是在 Web 服务器上配置的。更主要的是,B/S 应用模式突破了传统的 C/S 模式中计算机应用对局域网的局限,从而将计算机应用分布到整个互联网中,用户可以在任何地方登录 Web 服务器,按照用户角色,执行自己的业务程序。

在万维网中,通过 HTTP 协议实现客户端(浏览器)和 Web 服务器的信息交换,Web 的基本工作原理如图 1-1 所示。

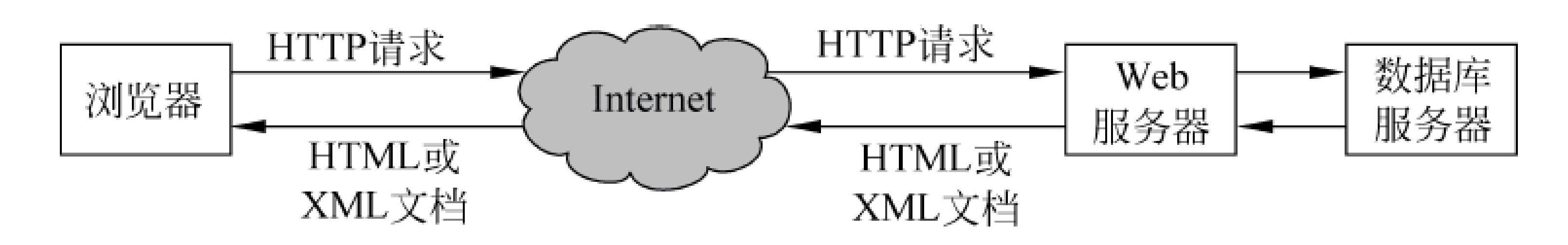



图 1-1 Web 的工作原理

在浏览器地址栏中,用户输入要访问的网页网址 URL(http://网址/路径/文件名.扩展名),向 Web 服务器提出 HTTP 请求。Web 服务器根据 URL 中指定的网址、路径和网页文件,调出相应的 HTML、XML 文档或 JSP、ASP 文件,根据文档的类型,Web 服务器决定是执行文档中的脚本程序,还是直接将网页文件传送到客户端。

现在一般的 Web 应用都是和数据库结合在一起的,服务器端脚本程序主要负责和数据库服务器建立连接并完成必要的数据查询、插入、删除、更新等数据库操作,然后利用获得的数据产生一个新的包含动态数据的 HTML 或 XML 文档,并将其发送给客户端 Web 浏览器。最后由 Web 浏览器解释该文档,在浏览器窗口中显示给用户。

### 1.2.3 浏览器

浏览器(Browser)就是前面经常提到的 Web 客户端程序,用户要浏览 Web 页面必须在本地计算机上安装浏览器软件。通过在浏览器地址栏中输入 URL 资源地址,将 Web 服务器中特定的网页文件下载到客户端计算机中,并在浏览器打开。因此,从本质上讲,浏览器是一种特定格式的文档阅读器,它能够根据网页内容,对网页中的各种标记进行解释显示。此外,浏览器又是一种程序解释机,如果网页中包含客户端脚本程序,浏览器将执行这些客户端脚本代码,从而增强网页的交互性和动态效果。不同版本的浏览器都需要遵循 HTML 规范中定义的标记集,同时为了便于脚本编程,每个浏览器程序本身也提供了相应的浏览器内置对象,类似于传统软件开发中的函数库及其标准库函数。

在 Web 发展初期,浏览器程序主要分为两类。一类是以 Lynx 为代表的基于字符的 Web 客户机程序,主要在不具备图形图像功能的计算机上使用。Lynx 是由美国堪萨斯大学的 Lou Montulli 研制的,同类的还包括 CERN 的 LineMode Browser。另一类是以 NCSA(National Center of Supercomputing Application)Mosaic 为代表的面向多媒体计算机的 Web 客户机程序,它可以在各种类型的小型机上运行,也可以在 IBM PC、Macintosh 机以及 UNIX 操作系统软件平台上运行。目前,使用最多的浏览器是微软的 IE(Internet Explorer)浏览器。除此之外,一些新的浏览器产品也不断推向市场,例如:Maxthon(遨游)、Firefox(火狐狸)、Opera 等。

## 1.3 概念及术语

在 Web 中,新的概念、术语很多,随着 Web 应用的普及,这些本来是专业的概念和术语已经大众化了,下面从计算机专业的角度对 Web 中的一些常用概念进行简要介绍。

#### 1. 网站(Web Site)

网站又称 Web 站点,是 Internet 中提供信息服务的机构,这些机构的计算机连接到 Internet 中,向用户提供 Web 服务。

从技术上讲,一个 Web 站点是由一个主目录、主目录下的文件夹和大量的网页文件构成的,这些网页文件通过超链接连接在一起,形成特定的应用逻辑,构成一个特定的 Web 应用。因此,网站又称为 Web 应用。

#### 2. 超文本(Hypertext)

超文本是一种文本显示与连接技术,可以对文本中的有关词汇或句子建立链接(即超链接),使其指向其他段落、文本或链接到其他文档。通过超链接,可以在文档之间、文档内部之间跳转,这种文本的组织方式与人们的思维方式和工作方式比较接近。

当超文本显示时,建立了链接的文本、图片通常以下划线、高亮等不同的方式显示,来表明这些文本或图片对应一个超链接。当鼠标移过这些文字时,鼠标会变成手形,单击超链接文本或图片,可以转到相关的位置。

#### 3. 超级链接(Hyperlink)

Web 页中当用户单击它时可以转到其他 Web 页或当前页面的其他地方的文字、图片等对象,即文本超链接和图片超链接。如果是文本超链接,超链接在 Web 页上往往带有下划线或增亮显示。当用户将鼠标指向一个超链接时,鼠标指针会改变为手的形状。

#### 4. Web 页(Web Page)

Web 页是指 Web 服务器上的一个个超文本文件,或者是它们在浏览器上的显示屏幕。Web 页中往往包含指向其他 Web 页面的超级链接。

#### 5. 主页(Home Page)

用户在 Web 服务器上看到的第一个 Web 页,该 Web 页一般的名称为 default. htm 或 index. htm。首页中往往列出了网站的信息目录,或指向其他站点的超链接,主页是一个网站的入口。当用户访问一个网站时,如果在 URL 中不指定特定的网页文件,则 Web 服务器将站点首页发送到客户端。

#### 6. 统一资源定位器 URL(Uniform Resource Locator)

统一资源定位器可以唯一标识一个 Web 页或 Internet 上其他资源的一个地址,它将 Internet 提供的各类服务统一编址,以便用户通过 Web 客户浏览程序进行信息查询。

URL 的一般形式为:

```
信息资源类型://网址:端口号/文件路径/文件名?参数表
```

其中,信息服务类型主要包括 http、ftp 等;网址即服务器的域名或 IP 地址,端口号对应一个特定的服务,默认端口号可以省略,例如 Web 服务的默认端口为 80,FTP 服务的默

认端口为21等；文件路径为网页相对于主目录的相对路径；文件名是用户浏览器指定的要下载的网页文件；如果有参数,在文件名后面跟字符“?”列出参数名及实际参数值。

在浏览器地址栏URL中,默认端口号可以省略不写,如果不指定文件路径和文件名,则默认访问站点根目录下的首页文件,首页文件由Web服务器指定。例如,如果用户在浏览器中输入的URL为：http://www.sdu.edu.cn,则表明用户要下载域名为www.sdu.edu.cn的Web服务器中根目录下的首页文件。

#### 7. 端口(Port)

端口是服务器使用的一个通道,在OSI参考模型的数据封装中使用,它可以使具有相同IP地址的服务器同时提供多种服务,运行在服务器上的各个服务程序将根据收到的数据段中的端口号来判断是否为自己的数据。例如,在一台计算机上同时提供WWW服务和FTP服务,WWW服务使用端口80,FTP服务使用端口21等。在通用资源定位器URL中,默认端口号可以省略不写。

另外,在一台计算机上,在同一个IP地址下,可以建立多个Web站点,站点之间用端口号来区分,访问不同的站点必须指定特定的端口号。

#### 8. 下载(Download)

下载是指通过Internet将文件从Web服务器或FTP服务器端传输到本地计算机的过程。

#### 9. 上传(Upload)

上传是指通过Internet将文件从本地计算机传输到Web服务器或FTP服务器计算机的过程。

## 1.4 Web相关技术

进入20世纪90年代以后,随着Internet技术的不断发展,特别是Web的出现,对计算机的计算模式、软件开发模式、应用模式都产生了重要的影响,这导致了一系列相关技术的出现,并推动着Web技术的发展。

### 1.4.1 浏览器/服务器计算模式

在Web出现以前,计算机的应用模式经历了单机应用到网络应用两个阶段,这些不同的计算模式有各自的优点和不足。Web的出现使得一种围绕Web服务的计算模式成为当前计算机应用的主流模式,并推动了软件开发、软件应用、应用集成方式上的重大改变。

#### 1. 集中式计算模式

在计算机诞生和应用的初期,计算所需要的数据和程序都是集中在一台计算机上进行

的，称为集中式计算。随着网络的发展，这种集中式计算往往形成一种由大型机和多个与之相连的终端组成的网络结构。当支持大量用户时，大型机自顶向下的维护和管理方式显示出集中式处理的优越性。它具有安全性好、可靠性高、计算能力和数据存储能力强以及系统维护和管理的费用较低等优点。但是它也存在着一些明显的缺点，如：大型机的初始投资较大、可移植性差、资源利用率低以及网络负载大等。

### 2. 客户/服务器(C/S)计算模式

随着微型计算机和网络的发展，数据和应用逐渐转向了分布式，即数据和应用程序跨越多个结点机，形成了新的计算模式，这就是客户/服务器计算模式(Client/Server，C/S)。C/S模式是一种典型的两层计算模式，它将应用一分为二：前端是客户机，一般使用微型计算机，几乎所有的应用逻辑都在客户端运行和表达，客户机完成与用户的交互任务，具有强壮的数据操纵和事务处理能力；后端是服务器，可以使用各种类型的主机，服务器负责数据管理，提供数据库的查询和管理、大规模的计算等服务。

C/S计算模式具有以下几个方面的优点：通过异种平台集成，能够协调现有的各种IT基础结构；分布式管理；能充分发挥客户端PC的处理能力，安全、稳定、速度快，且可脱机操作。但随着应用规模的日益扩大，应用程序的复杂程度不断提高，C/S结构逐渐暴露出许多的缺点和不足，主要包括：它必须在客户端安装大量的应用程序(客户端软件)，开发成本较高，移植困难，用户界面风格不统一，使用繁杂，不利于推广使用，维护复杂，升级麻烦，信息内容和形式单一，新技术不能轻易应用等。

### 3. 浏览器/服务器(B/S)计算模式

客户/服务器模式表现出了许多不足，特别是它的胖客户机和对局域网的依赖，已经不能适应Web的发展。人们需要利用互联网，将应用分布到整个Web中，而不是局限于企业局域网内部，这就导致了一种更加灵活的多级分布式计算模式，即浏览器/服务器模式(Browser/Server，B/S)的产生和发展。

浏览器/服务器(B/S)计算模式是一种基于Web的协同计算，是一种三层架构瘦客户机/服务器计算模式。第一层为客户端表示层，与C/S结构中的“肥”客户端不同，三层架构中的客户层只保留一个Web浏览器，不存放任何应用程序，其运行代码可以从位于第二层的Web服务器下载到本地的浏览器中执行，几乎不需要任何管理工作，是一种“瘦”客户机。第二层是应用服务器层，由一台或多台Web服务器组成，处理应用中的所有业务逻辑、对数据库的访问等工作。该层具有良好的可扩充性，程序的部署和管理主要在Web服务器上进行，相对于C/S而言无论是工作的复杂性还是工作量都大大减少。第三层是数据中心层，安装数据库服务器，负责整个应用中的数据管理。

B/S计算模式与传统的C/S结构相比体现了集中式计算的优越性：具有良好的开放性，利用单一的访问点，用户可以在任何地点使用系统；用户可以跨平台以相同的浏览器界面访问系统；因为在客户端只需要安装浏览器，基本上取消了客户端的维护工作，有效地减少了整个系统的运行和维护成本。

### 1.4.2 Java 技术

Java 技术是 Sun Microsystems 于 1995 年推出的一种极富创造力的计算平台。狭义上讲,Java 技术可以理解为 Java 语言,广义上讲,Java 技术包括 Java 语言、Java 虚拟机以及 Java API 等。Java 技术为用户带来了无数令人兴奋的可能性,它几乎使所有应用程序(包括游戏、工具及信息程序和服务)都能在任何计算机或设备上运行。Java 技术的多功能性、有效性、平台的可移植性以及安全性已经使它成为网络计算领域最完美的技术。今天,Java 技术已经无处不在,从桌面 PC 到科学超级计算机和互联网,从移动电话到移动手持设备,从家庭游戏机到信用卡,几乎在所有的网络和设备上都会看到 Java 技术的身影。

#### 1. Java 的出现

1991 年,Sun 计划开拓消费类电子产品市场,为电视、烤面包箱等家用消费类电子产品开发一个分布式代码系统,目的是可以通过 Internet 与家电产品进行交互,以便对其进行控制。Sun 内部人员把这个项目称为 Green,该小组的领导人是 James Gosling,他是一位非常杰出的程序员。Gosling 于 1984 年加盟 Sun Microsystem 公司,之前在一家 IBM 研究机构工作。他是 SunNeWs 窗口系统的总设计师,也是第一个用 C 实现的 EMACS 文本编辑器 COSMACS 的开发者。

开始,他们准备用 C++语言开发,但是,C++太复杂,且存在安全性问题。于是在 1991 年 6 月 James Gosling 开始准备基于 C++开发一个新的语言,他看着窗外的一棵老橡树,就将这个新的语言命名 Oak,它就是 Java 的前身。Oak 是一种用于网络的精巧而安全的语言,Sun 使用它参加一个交互式电视项目的投标,结果败于 SGI①,为此 Oak 几乎销声匿迹。此时,受到 Mark Ardreesen 开发的 Mosaic 和 Netscape 的启发,他们将 Oak 继续完善。因为此时发现在此之前 Oak 已是 Sun 公司另一个语言的注册商标,他们将新的 Oak 改名为 Java,即太平洋上一个盛产咖啡的岛屿(爪哇岛)的名字。

James Gosling 在开始写 Java 时,并不局限于扩充语言机制本身,更注重于语言所运行的软硬件环境。他要建立一个系统,这个系统运行于一个巨大的、分布的、异构的网格环境中,完成各种电子设备之间的通信与协同工作。Gosling 在设计中采用了虚机器码(Virtual Machine Code)方式,即 Java 语言编译后产生的是虚拟机,虚拟机运行在一个解释器上,每一个操作系统均有一个解释器。这样一来,Java 就成了平台无关语言,这和 Gosling 设计的 SunNeWs 窗口系统有着相同的技术味道。在 NeWs 中用户界面统一用 Postscript 描述,不同的显示器有不同的 Postscript 解释器,这样便保证了用户界面的良好的可移植性。

后来,Patrick Naughton 加入到该项目,Naughton 也是 Sun 公司的技术骨干,曾经是 Open Windows 项目的负责人。整个工作进展神速,经过 17 个月的奋战,整个系统胜利完成。它是由一个操作系统、一种语言(Java)、一个用户界面、一个新的硬件平台、三块专用芯片构成的。项目完成后,在 Sun 公司内部做了一次展示和鉴定,观众的反应是:在各方面都采用了崭新的、非常大胆的技术。

① SGI 公司为美国图形工作站生产厂商。

### 2. Java 语言环境

1994 年，WWW 已如火如荼地发展起来。Gosling 意识到 WWW 需要一个中性的浏览器，它不依赖于任何硬件平台和软件平台，它应是一种实时性较高、可靠安全、有交互功能的浏览器。于是 Gosling 决定用 Java 开发一个新的 Web 浏览器。这项工作由 Naughton 和 JonathanPayne 负责，到 1994 年秋天，完成了 WebRunner 的开发工作。WebRunner 是 HotJava 的前身，这个原型系统展示了 Java 可能带来的广阔市场前景。WebRunner 改名为 HotJava，并于 1995 年 5 月 23 日发表后，在产业界引起了巨大轰动，Java 的地位也随之而得到肯定。

在 1995 年 Sun 虽然推出了 Java，但这只是一种语言，而要想开发复杂的应用程序，必须要有一个强大的开发库支持。因此，又经过一年的试用和改进，Sun 在 1996 年 1 月 23 日发布了 JDK1.0。这个版本包括了两部分：运行环境（即 JRE）和开发环境（即 JDK）。在运行环境中包括了核心 API、集成 API、用户界面 API、发布技术、Java 虚拟机（JVM）五个部分。开发环境包括编译 Java 程序的编译器（即 javac）。在 JDK1.0 时代，JDK 除了 AWT（Abstract Windowing Toolkit，抽象窗口工具包，一种用于开发图形用户界面的 API）外，其他的库并不完整。

在推出 JDK1.0 后，紧跟着，Sun 在 1997 年 2 月 18 日发布了 JDK1.1。JDK1.1 相对于 JDK1.0 最大的改进就是为 JVM 增加了 JIT（即时编译）编译器。JIT 和传统的编译器不同，传统的编译器是编译一条，运行完后再将其扔掉，而 JIT 会将经常用到的指令保存在内存中，在下次调用时就不需要再编译了，这样 JDK 在效率上有了非常大的提升。

随后一些著名的计算机公司纷纷购买了 Java 的使用权，IBM、Apple、DEC、Adobe、Silicon Graphics、HP、Oracle、Toshiba、Netscape 和 Microsoft 等大公司相继购买了 Java 的许可证。另外，众多的软件开发商也开发了许多支持 Java 的软件产品。在以网络为中心的计算时代，不支持 HTML 和 Java，就意味着应用程序的应用范围只能限于同质的环境。

Java 的平台无关性给未来的计算模式产生了革命性的影响，它是继 HTML 后，Internet 发展的又一个里程碑。

### 3. Java 的技术特征

在 Sun 的 Java 语言白皮书中，说明 Java 语言有如下特征：简单、面向对象、分布式、解释执行、健壮、安全、体系结构中立、可移植、高性能、多线程、动态性等。

1）简单（Simple）

主要体现在三个方面：①Java 语言风格来源于 C++，因此 C++程序员可以很快的上手。②Java 摒弃了 C++中容易引发错误的地方，如：指针，增加了内存管理等一些新的特色。③Java 提供了丰富的类库，使用户编程更加简单。

2）面向对象（Object-Oriented）

Java 是面向对象的语言，摒弃了 C++中全局变量等与面向对象思想冲突的内容。

3）体系结构中立（Architecture Neutral）

一般情况下，网络环境都是异构的，如何使一个应用程序能够在不同硬件、不同操作系统平台的计算机上运行，始终是一个难题。Java 将它的程序编译成一种结构中立的中间文

件格式，由Java虚拟机来解释执行这种中间代码。这使得Java应用程序可以在不同的处理器中执行，现在几乎所有的主流计算机系统都能运行Java。

4）解释执行（Interpreted）

Java解释器能直接地在任何机器上执行Java字节码（Byte codes）。

5）可移植（Portable）

同体系结构无关的特性使Java程序可以在配备了Java虚拟机的任何计算机系统上运行。另外，通过定义独立于平台的基本类型及其运算，Java数据得以在任何硬件平台上保持一致。

6）分布式（Distributed）

Java程序的程序库可以很容易地与HTTP和FTP等TCP/IP协议配合，从而使Java程序可以凭借URL打开并访问网络对象，对程序员来讲，访问方式和访问本地文件系统几乎一样，这就为Internet等分布环境提供内容带来了方便。

7）安全性（Secure）

Java是被设计用于网络和分布式环境的，安全性自然是一个重要的考虑因素。Java的安全性可以从两个方面考虑：①内存的安全性，如摒弃了C++中的指针，从而避免了非法内存操作和内存泄露。②当用Java来创建浏览器内容时，语言功能和浏览器本身的功能结合，使它更安全。

#### 4. Java的发展

十多年来，Java技术的发展总是日新月异，从奠定Java根基的Java开发包JDK1.0到今天的JDK6，Java为开发人员提供的标准类库越来越丰富，Java技术取得了长足的进步。

从JDK1.0到JDK1.1.8，JDK1.x经过了9个小版本的发展，已经初具规模。在1998年12月4日，Sun发布了Java历史上最重要的一个JDK版本：JDK1.2，这个版本标志着Java进入了Java2时代，进入Java的飞速发展时期。

在Java2时代，Sun对Java进行了很多革命性的变化，Sun将JDK1.2一分为三，Java被分成了J2EE（Java2 Platform，Enterprise Edition）、J2SE（Java2 Platform，Standard Edition）和J2ME（Java2 Platform，Micro Edition），分别面向企业级、桌面、嵌入式和移动计算等领域。这些革命性的变化一直沿用到现在，对Java的发展形成了深远的影响。

从JDK1.2开始，Sun以平均两年一个版本的速度推出新的JDK。在2000年5月8日，Sun对JDK1.2进行了重大升级，推出了JDK1.3。Sun在JDK1.3中同样进行了大量的改进，主要表现在一些类库上（如数学运算、新的Timer API等），在JNDI接口方面增加了一些DNS的支持，增加了JNI的支持等。2002年2月13日，Sun发布了JDK历史上最为成熟的版本JDK1.4。这次Sun将主要精力放到了Java的性能上，使JDK1.4的性能有了质的飞跃。到JDK1.4为止，我们已经可以使用Java实现大多数的应用了。

虽然从JDK1.4开始，Java的性能有了显著的提高，但Java又面临着另一个问题，那就是复杂。在2004年10月，Sun发布了JDK1.5，同时，Sun将JDK1.5改名为J2SE5.0。和JDK1.4不同，JDK1.4的主题是性能，而J2SE5.0的主题是易用。

2006年4月，Sun推出J2SE6.0测试版，2006年12月，代号为Mustang（野马）的J2SE6.0正式版推向市场，在性能、易用性方面得到了前所未有的提高。

### 1.4.3　XML 技术

可扩展标记语言 XML(eXtensible Markup Language)是 Internet 上最具权威的数据表示和数据交换标准，它是 ISO(International Organization for Standardization，国际标准化组织)的 SGML(Standard for General Markup Language，通用标记语言标准)的一个简化子集。

#### 1. XML 的技术特征

XML 关注信息本身，是 Web 上表示结构化信息的一种标准文本格式。与传统的注重页面信息显示的 HTML(Hypertext Markup Language，超文本标记语言)相比，关注于内容的 XML 具有以下诸多优点：良好的可扩展性，语言简单有效，可自行定义标记；内容与形式的分离，主要刻画数据内容，不考虑显示效果；有严格的语法要求，便于分析和与数据库信息转换；便于传输，为纯文本形式，可通过 Http 协议直接传输，可跨越防火墙等。

XML 的出现和发展对于 Internet 产生了巨大的影响，如果说 Java 实现了代码的平台无关性，那么 XML 则实现了数据的平台无关性。今天，XML 已经逐渐成为整个 Web 的基本结构和未来各种发展的基础，由于 XML 能针对特定的应用定义自己的标记语言，这一特征使得 XML 可以在电子商务、政府部门、各行业领域提供各具特色的独立解决方案。同时，XML 作为一种通用的数据交换语言，已经成为业界的一种具有垄断性的标准，在跨平台跨系统数据交换方面拥有无可比拟的优势，其在企业级开发中所扮演的角色越来越重要。但是，和关系数据库拥有强大的存储和分析引擎不同，XML 只专注于数据的表示，这也使得 XML 在数据量急速膨胀的时候，如何有效地管理和使用 XML 成为了一件令人头痛的事情。

#### 2. XML 相关技术标准

虽然 XML 标准本身相对简单，但与 XML 相关的标准却种类繁多，W3C 制定的相关标准就有二十多个，采用 XML 制定的各种应用标准也很多。除了标准种类繁多外，标准之间通常还互相引用，特别是应用标准，它们的制定不仅仅使用的是 XML 标准本身，还常常用到了其他很多标准。在 XML 标准体系中，XML 相关标准可分为元语言标准、基础标准、应用标准三个层次。

(1) 元语言标准(meta-language)：描述的是用来描述标准的元语言，在 XML 标准体系中就是 XML 标准，是整个体系的核心，其他 XML 相关标准都是用它制定的或为其服务的。

(2) 基础标准(Foundation Standards)：这一层次的标准是为 XML 的进一步实用化制定的标准，规定了采用 XML 制定标准时的一些公用特征、方法或规则。如：XML Schema 描述了更加严格地定义 XML 文档的方法，以便可以更自动化地处理 XML 文档；XML Namespace 用于保证 XML DTD 中名字的一致性，以便不同的 DTD 中的名字在需要时可以合并到一个文档中。

(3) 应用标准(Application Standards)：以 XML 为基础制定的行业标准。比较常用的应用标准包括：SVG(有关矢量图形)、SMIL(有关多媒体同步显示)、MathML(有关数学公

式符号)等。在电子商务领域的常用应用标准有:Micropayments(W3C 制定)、BizTalk(Microsoft 发起的电子商务的 schema 库)、ebXML(联合国 UN/CEFACT 小组和 OASIS 共同发起)、PIP(由诸多 IT 业的巨子组成的一个标准化组织 RosettaNet 的应用网络标准),等等。

XML 相关技术标准体系如图 1-2 所示。

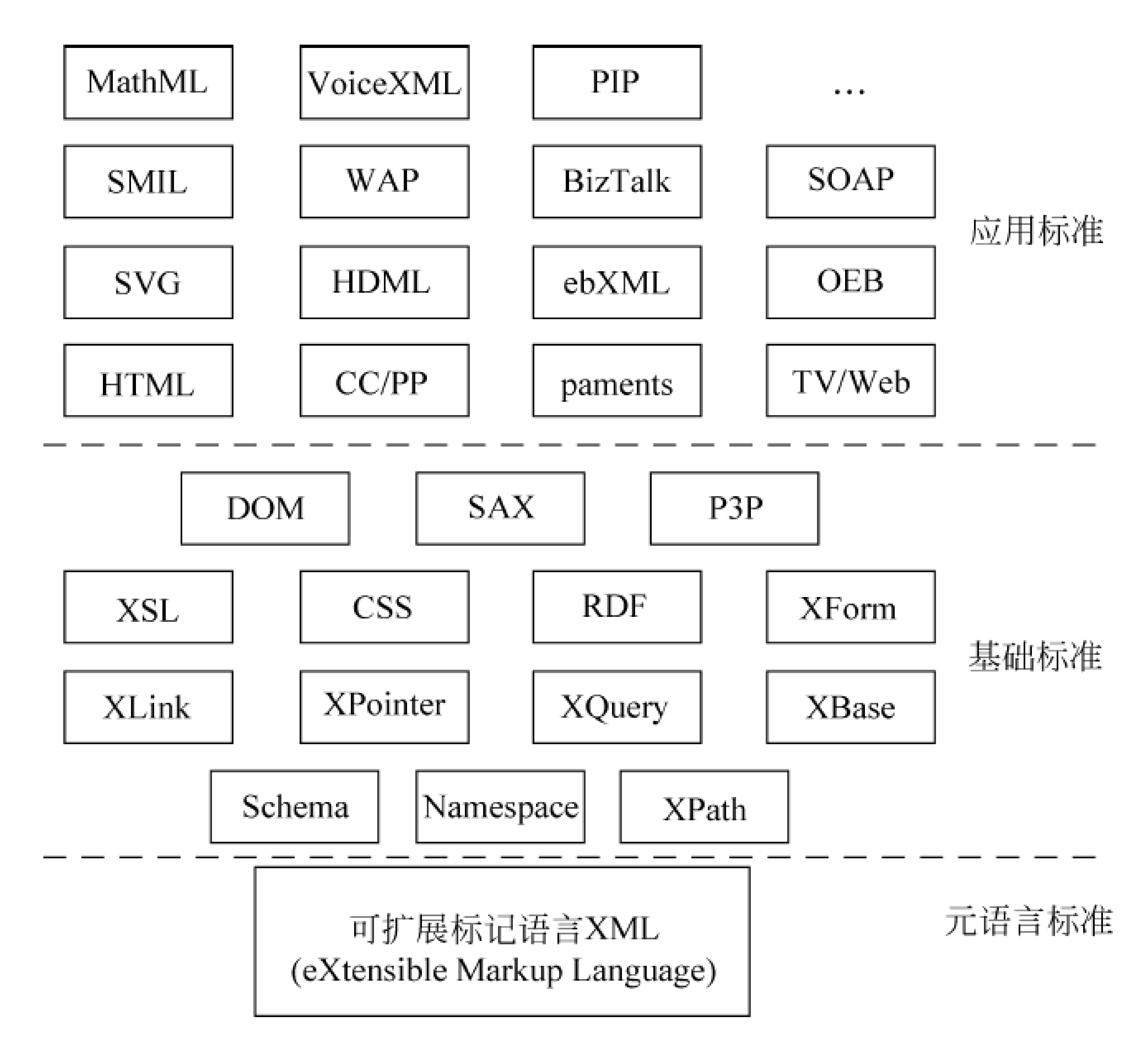



图 1-2 XML 技术标准体系

在 XML 技术标准体系中,主要的基础标准的功能划分是:

XML Schema 描述了更加严格定义 XML 文档的方法,以便可以更自动地处理 XML 文档。XML Namespace 用于保证 XML DTD 中名字的一致性,以便不同的 DTD 中的名字在需要时可以合并到一个文档中。XPath 描述如何识别、选择、匹配 XML 文件中的各个构成元件,包括元素、属性、文字内容等。XPointer 和 XLink 标准规定了有关定位、链接方面的内容。XQuery 的目的是为从 Web 文档中提取数据提供一种灵活的查询机制。XSLT 则实现文档格式转换,主要是将 XML 转换为 HTML 格式进行显示。CSS 也是用来作为 XML 文档显示的样式标准。DOM 定义了一组与平台和语言无关的接口,以便程序和脚本能够动态访问和修改 XML 文档内容、结构及样式。

### 1.4.4 Web 服务

Web 服务(Web Service)是在 XML 技术的基础上发展起来的,它是可以通过 Web 发布、查找和调用的自包含、自描述的模块化应用,其目标是将软件转化为一种可以通过 Web 订阅使用的服务。

1. Web服务体系架构

Web 服务使用标准化的 XML 消息传递机制作为基本的数据通信方式，消除使用不同组件模型、操作系统和编程语言的系统之间存在的差异，使异类系统能够作为计算网络的一部分协同运行。开发人员可以使用像过去创建分布式应用程序时使用组件的方式，创建由各种来源的 Web 服务组合在一起的应用程序。

Web 服务的体系结构由三个参与者和三个基本操作构成。三个参与者分别是服务提供者、服务请求者和服务代理，三个基本操作分别为发布(publish)、查找(find)和绑定(bind)，其关系如图 1-3 所示。

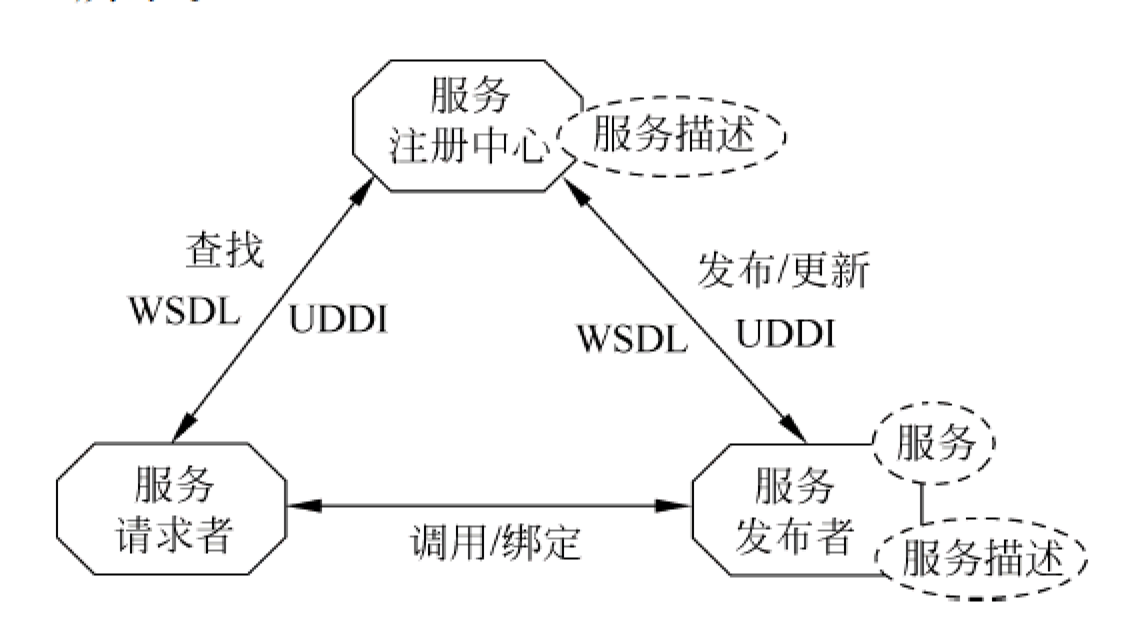



图 1-3　Web 服务体系结构

服务提供者将其服务发布到服务代理的一个目录上；当服务请求者需要调用该服务时，他首先利用服务代理提供的目录去搜索该服务，得到如何调用该服务的信息；然后根据这些信息去调用服务提供者发布的服务。当服务请求者从服务代理得到调用所需服务的信息之后，通信是在服务请求者和提供者之间直接进行，而无须经过服务代理。

Web 服务主要基于以下技术标准。

(1) 可扩展标记语言 XML：它是 Web 服务的驱动力，它不是一种编程语言或者 API，而是一种独立于平台的组织数据的方式。XML 的语法便于通过编程来处理文本数据，同时又便于为人们所理解。Web 服务使用 XML 作为标准，在网络设备之间进行通信。

(2) 简单对象访问协议 SOAP(Simple Object Access Protocol)：开发人员可以使用这种独立于平台的机制，远程调用分布式对象的方法。SOAP 消息的通信使用 XML 来描述对象、方法以及执行的参数。客户机和服务器都可以实现和使用 SOAP。

(3) Web 服务描述语言 WSDL(Web Service Description Language)：它从句法层面对 Web 服务的功能进行描述，包括 4 个不同的粒度：数据类型(Data type)、消息(Message)、方法(Operation)和访问端口(Port Type)。WSDL 只是提供了 Web 服务的接口描述，对服务的行为约束和属性描述缺乏进一步的支持。

(4) 语义 Web 服务标记语言 OWL-S：OWL-S 是语义 Web 服务标记语言的标准，它比 WSDL 更能向用户提供可理解的服务资源的描述形式，提高服务选取与推荐的准确性。语义 Web 服务的主要方法是利用本体(Ontology)来描述 Web 服务，然后通过这些带有语义信息的描述实现 Web 服务来实现服务的自动发现、调用和组合。语义 Web 和 Web 服务是语义 Web 服务的两大支撑技术。OWL-S 是连接两大技术的桥梁，目前对语义 Web 服务标

记语言研究最重要的组织就是 DARPA 组织,其研究组 OWL Services Coalition 提出了语义 Web 服务标记语言 OWL-S。

(5) 通用描述发现和集成协议 UDDI(Universal Description,Discovery and Integration):提供了一组公用的 SOAP API,使得服务代理得以实现。UDDI 为发布服务的可用性和发现所需服务定义了一个标准接口(基于 SOAP 消息)。UDDI 实现将发布和发现服务的 SOAP 请求解释为用于基本数据存储的数据管理功能调用。

为了发布和发现其他 SOA 服务,UDDI 通过定义标准的 SOAP 消息来实现服务注册(Service Registry)。注册是一种服务代理,它是在 UDDI 上需要发现服务的请求者和发布服务的提供者之间的中介。一旦请求者决定使用特定的服务,开发者通常借助于开发工具(如 Microsoft Visual Studio. NET)并通过创建以发送请求并处理响应的方式访问服务的代码来绑定服务。

#### 2. Web 服务技术优势

总结 Web 服务技术,有如下技术优势:

(1) 平台无关、语言无关性。Web 服务技术的主要目标是在现有的各种异构平台的基础上构筑一个通用的平台无关、语言无关的技术层,各种不同平台之上的应用可以依靠这个技术层来实施彼此的连接和集成。

(2) 自描述能力。Web 服务的所有协议,包括 SOAP、WSDL、UDDI 都是 XML 文档,所以 Web 服务具有自描述的良好性质。

(3) 松耦合性。当一个 Web 服务的实现发生变更时,调用者是不会感到这一点的。对于调用者来说,只要 Web 服务的调用接口不变,Web 服务实现的任何变更对他们来说都是不透明的,甚至当 Web 服务的实现平台从 J2EE 迁移到. NET 或者是相反的迁移流程,用户都可以对此一无所知。

(4) 易于集成。Web 服务采用简单的、易理解的标准 Web 协议作为组件界面描述和协同描述规范,完全屏蔽了不同软件平台的差异,无论是 CORBA、DCOM 还是 EJB 都可以通过这一种标准的协议进行互操作,实现了在当前环境下最高的可集成性。

(5) 用消息传递代替传统的 APIs。Web 服务采用了 SOAP 协议。SOAP 协议独立于平台,可以在不同的平台、环境下进行传递和交互。

Web Service 是组件技术在 Internet 中的延伸,从本质上讲是放置于网络上的可重用构件。从更高的概念层面讲,可以将 Web 服务视为一些工作单元,每个单元处理特定的功能任务。再往上一步,可以将这些任务组合成面向业务的任务,以处理特定的业务操作任务,从而使非技术人员去考虑一些应用程序,这些应用程序可以在 Web 服务应用程序工作流中一起处理业务问题。因此,一旦由技术人员设计并构建好 Web 服务之后,业务流程架构设计师可以聚集这些 Web 服务来解决业务层面上的问题。

### 1.4.5 基于 SOA 的软件设计模式

从计算机诞生到现在,计算机硬件技术在发展的同时,计算机软件也在悄悄地发生变化。这种变化不仅表现在计算机软件的内涵、计算机软件的应用方式上,同时也表现在软件

的设计模式和开发方法上。软件开发思想的演化，使得在 Web 流行的今天，面向服务的体系架构 SOA 成为下一代软件体系架构的主流。

#### 1. 软件设计与开发模式的演化过程

站在软件开发人员的角度，我们往往希望软件开发能够满足对于开发效率、可靠性、易维护性、易管理等多方面的更高要求。无论在计算机发展的哪一个时期，这都是软件开发人员永恒的目标。我们可以把这种目标方法的实现分成以下几个阶段：

（1）面向机器语言的开发模式。在计算机诞生的初期，需要根据不同平台的机器语言来开发代码。这一阶段，程序规模很小，谈不上系统的方法。

（2）软件的生命周期开发模式，又称软件开发瀑布模型，是面向功能或过程的软件开发方法。它将软件开发分成以下几个阶段。

① 用户需求分析：开发人员和业务人员交流，制定用户需求说明书；

② 系统设计：开发人员根据需求说明书进行系统设计，制定系统设计报告；

③ 系统开发：根据系统设计报告，进行系统编码；

④ 系统测试：系统实现后双方组织人员进行测试；

⑤ 运行与维护：系统测试结束后，便进入系统的运行、维护期。

从理论上讲，软件开发的生命周期模式是非常科学的，但是利用生命周期模式开发系统基于两个假设：一是用户能够清楚地、完整地提供系统要求；二是开发者能完整地、严格地理解和定义要求。但在实际开发中，这两个假设是很难满足的。因为在开发初期，用户很难清楚地描述系统需求，或者系统需求将来可能发生较大变化。其次，开发人员和业务人员在交流时可能存在理解上的不一致，其结果是系统开发完毕后，不能很好地满足用户需求。生命周期模式是封闭式的，缺少灵活性，特别是在用户需求定义方面。

面向过程的生命周期法主要流行于 20 世纪 80 年代，直到现在，这种思想一直还对软件设计和开发人员有着很深的影响。

（3）原型法开发模式。和生命周期模式不同，原型法要求经过对用户需求的简单快速分析，利用高级开发工具及环境，快速完成原型系统的设计和实施，提供给用户评价。一个原型系统就是系统的一个可运行的早期版本，它反映了最终系统的部分重要特征，在评价过程中，开发人员不断从用户那里得到反馈信息，修正原型的用户需求定义，进而对原型系统作相应改进，逐步减少分析与交互过程中的误解，弥补遗漏，从而提高最终系统的质量。

原型法的核心是用交互的、快速建立起来的原型取代了形式的、不易修改的大块的规格说明，用户通过在计算机上实际运行和试用原型而向开发者提供真实的反馈意见。原型法开发模式的实现得益于面向对象的语言（Smalltalk、C++、Java 等）和可视化的第四代开发工具的出现。从宏观上讲，原型法比生命周期更实用，但是，在每一个原型的设计和开发过程中，都离不开生命周期的科学思想。

在软件工程的实践中，“生命周期法”和“原型法”的有效结合表现出了强大的生命力和可操作性。这种结合就是，整个软件的开发表现为一个个原型的向前推进，在每一个原型的内部，又是按照生命周期的思想来设计。

（4）面向组件（Component）的模式。随着软件开发规模的扩大，在涉及分布式、异构等复杂特征的环境中，代码级别的重用性差、可维护性差、效率低的弱点是不可逾越的，因此人

们以架构运行环境(如.NET、J2EE 等)来提供完善的支撑平台,从而把开发者解放出来,更专注于业务核心的开发。而这些业务功能(Business Function)以组件的形式(DCOM、EJB等)发布运行在架构运行环境中,软件开发的重用模式也上升到业务组件的级别。

(5) 面向方面编程(Aspect Oriented Programming,AOP)模式。AOP 是 OOP 的延续,设计模式孜孜不倦追求的是调用者和被调用者之间的解耦,AOP 可以说是这种目标的一种实现。AOP 使原本复杂的调用与被调用和错综复杂的耦合关系变得清晰,使程序的整体架构保持高内聚、低耦合,这对于一个大型复杂系统来说是非常重要的。

(6) 面向服务(SOA)的模式。当软件的使用范围扩展到更广阔的范围,往往会面对更加复杂的 IT 环境和更加灵活多变的需求。服务(Service)的概念出现了,人们将应用(Application)以业务服务(Business Service)的形式公布出来供别人使用,而完全不需要去考虑这些业务服务运行在哪一个架构体系上,因为所有的服务都讲着同样的语言。

SOA 同样也强调重用(Reuse),但是相对于传统的代码重用、对象重用和部件重用,SOA 的重用粒度更粗。SOA 的核心体现在企业应用或者业务功能上的"重用"和"互操作",SOA 的重用在于业务级的应用,即服务的重用,而不再把 IT 与业务对立起来,这可以被视为在 IT 驱动业务的方向上迈出的重要一步。

#### 2. SOA 设计思想

面向服务的体系架构 SOA(Service-Oriented Architecture)是软件体系架构的下一代发展方向。SOA 以可重用、模块化和松耦合为特征,将业务逻辑用服务体现出来。SOA 体系架构可以应用在应用集成和软件架构设计两个不同的层面。

首先,从设计思路上看,SOA 并不是一个新的概念,CORBA(通用对象请求代理体系架构)的出发点同样也是想实现类似的目标,但是由于在标准、中间件以及 API 的欠缺,使得 CORBA 技术只能纸上谈兵,而无法实现。今天,企业应用集成的迫切需求使得 SOA 再一次被提到前台,但是今天的 SOA 已经与以往 CORBA 时代所面临的环境大不相同了,其中最为核心的技术——Web 服务已经拥有了一系列标准,WSDL、UDDI 等标准都已经被融入到了各厂商的产品之中,为 SOA 的实现奠定了技术基础,使 SOA 技术的实现与应用成为可能,也成为 SOA 软件集成技术快速发展的动力,使 SOA 进入主流软件集成技术体系。

此外,SOA 技术的逐步成熟,不仅将深刻改变软件集成的体系架构和工作方法,同时,也将彻底改变软件开发的思想、架构和工作方法。

SOA 是一种思想、模式和体系。它规范了在软件架构以及系统集成中的方法,其思想的表现就是将业务逻辑和功能分解成更小的独立逻辑和功能单元。通过聚合技术,将这些单元构建成一个较大的业务逻辑单元,从而实现服务的独立存在,通过标准技术,使服务保持足够的共性,实现系统的体系化。

下面是一个人才评价服务支撑平台的 SOA 体系架构,如图 1-4 所示。

上述 SOA 体系架构反映出基于 SOA 的软件设计模式和设计思想与传统的软件体系架构的不同。在该体系架构下,利用服务管理工具集、业务流程管理工具集,把最基本的共性业务抽象出来,封装为一个个基础服务,建立基础共性服务库。利用业务流程管理工具集,实现业务流程的定制和编排。完成后的服务在服务容器中,通过服务总线进行协同。

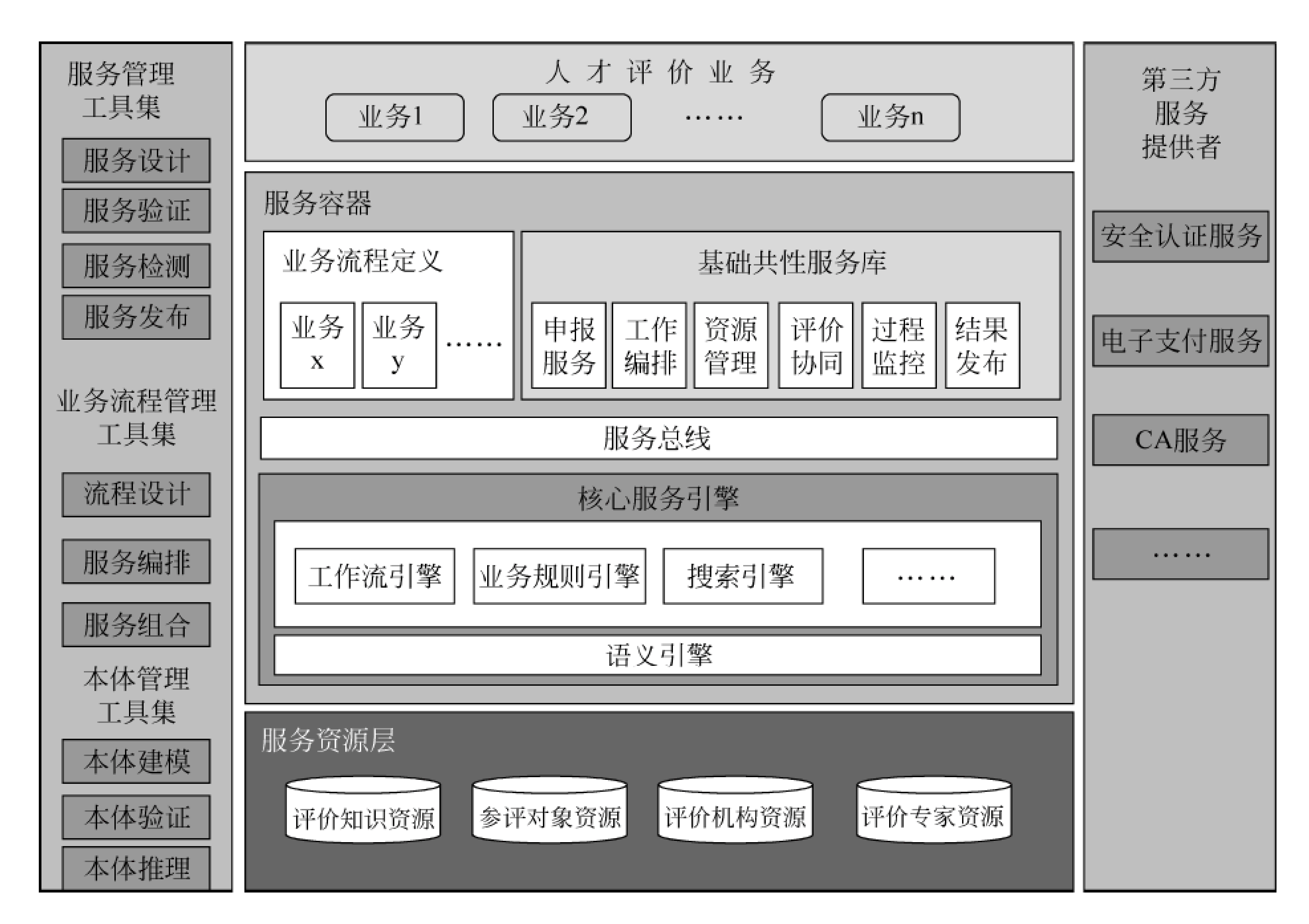



图 1-4　某人才评价服务支撑平台 SOA 体系架构

### 3. SOA 的特征

面向服务的体系架构 SOA 具有如下特征：

（1）服务的封装(Encapsulation)。把服务封装成可以被不同业务流程重复使用的业务组件。它隐藏所有实现细节，不管服务内部如何修改，使用什么平台、什么语言，只要保持接口不变，就不会影响最终用户的使用。

（2）服务的重用(Reuse)。一个服务是一个独立的实体，与底层实现和用户的需求完全无关，极大地方便了服务的重复使用，从而降低了开发成本。

（3）服务的互操作(Interoperability)。互操作并不是一个新概念。在 CORBA、DCOM、Web Service 中就已经采用互操作技术了。在 SOA 中，通过服务之间既定的通信协议进行互操作。主要有同步和异步两种通信机制。SOA 提供服务的互操作特性更利于其在多个场合被重用。

（4）服务是自治的(Autonomous)功能实体。服务是由组件组成的组合模块，是自包含和模块化的。SOA 非常强调架构中提供服务的功能实体的完全独立自主的能力。传统的组件技术，如.NET Remoting、EJB、COM 或者 CORBA，都需要有一个宿主(Host 或者 Server)来存放和管理这些功能实体；当这些宿主运行结束时这些组件的寿命也随之结束。这样当宿主本身或者其他功能部分出现问题的时候，在该宿主上运行的其他应用服务就会受到影响。

在 SOA 架构中，非常强调实体自我管理和恢复能力。常见的用来进行自我恢复的技术，比如事务处理(Transaction)、消息队列(Message Queue)、冗余部署(Redundant

Deployment)和集群系统(Cluster)在 SOA 中都起到至关重要的作用。

(5) 服务之间的松耦合度(Loosly Coupled)。服务请求者到服务提供者的绑定与服务之间应该是松耦合的。这就意味着,服务请求者不知道提供者实现的技术细节,比如程序设计语言、部署平台等。服务请求者往往通过消息调用操作请求消息和响应,而不是通过使用 API 和文件格式。这个松耦合使会话一端的软件可以在不影响另一端的情况下发生改变,前提是消息模式保持不变。

(6) 服务是位置透明的(Location Transparency)。服务是针对业务需求设计的,需要反应需求的变化,即所谓敏捷(Agility)设计。要想真正实现业务与服务的分离,就必须使得服务的设计和部署对用户来说是完全透明的。也就是说,用户完全不必知道响应自己需求的服务的位置,甚至不必知道具体是哪个服务参与了响应。

SOA 的灵活性将给企业带来巨大的好处。如果把企业的 IT 架构抽象出来,将其功能以粗粒度的服务形式表示出来,每种服务都清晰地表示其业务价值,那么,这些服务的顾客就可以得到这些服务,而不必考虑其后台实现的具体技术。

基于以上特征,只要将 Web Service 置入特定的企业服务总线,就可以任意使用它。

#### 4. SOA 生命周期

由于 SOA 涉及到业务的诸多方面,因此需要从一开始就对 SOA 项目进行细心的规划和设计。

(1) 建模(Model):SOA 项目的第一步几乎和技术没有任何关系,所有事项都与用户的业务相关。面向服务的方法将业务所执行的活动视为服务,因此第一步是要确定这些业务活动或流程实际是什么。对用户的业务体系结构进行记录,这些记录不仅可以用于规划 SOA,还可以用于对实际业务流程进行优化。通过在编写代码前模拟或建模业务流程,可以更深入地了解这些流程,从而有利于构建帮助执行这些流程的软件。

建模业务流程的程度将依赖于预期实现的深度。对于企业架构师,需要对实际的业务服务进行建模。对于软件开发人员,可能对单个服务进行建模。

(2) 组装(Assemble):对业务流程进行了建模和优化后,开发人员可以开始构建新的服务和/或重用现有的服务,然后对其进行组装以形成组合应用程序,从而实现这些流程。在建模步骤中,已经确定了需要何种类型的服务以及它们将访问何种类型的数据。已经存在某种形式的实现这些服务或访问该类数据所需的一些软件。组装步骤将要找到已经存在的功能,并为其添加服务支持。另外,还涉及到创建提供功能和访问数据源所需的新服务,以便满足用户的 SOA 涉及的业务流程范围内的需求。

(3) 部署(Deploy):进行了建模和组装后,要将组成 SOA 的资产部署到安全的集成环境中。此环境本身提供专门化的服务,用于集成业务中涉及的人员、流程和信息。这种级别的集成可帮助确保将公司的所有主要元素连接到一起协同工作。此外,部署工作还需要满足业务的性能和可用性需求,并提供足够的灵活性,以便吸纳新服务,而不会对整个系统造成大的影响。

(4) 管理(Manage):部署后,需要从 IT 和业务两个角度对系统进行管理和监视。在管理步骤中收集的信息用于帮助实时地了解业务流程,从而能更好地进行业务决策,并将信息反馈回生命周期,以进行持续的流程改进工作。在这期间,将需要处理服务质量、安全、一般

系统管理之类的问题。

在本步骤中,将监视和优化系统,发现和纠正效率低下的情况和存在的问题。由于SOA是一个迭代过程,因此,在此步骤中,不仅要找出技术体系结构中有待改进之处,而且还要找出业务体系结构中有待改进之处。

完成此步骤后就要开始新的"建模"步骤了。在"管理"步骤中收集的数据将用于重复整个SOA生命周期,再次进行整个过程。

(5) 控制(Governance & Processes):SOA是一种集中系统,其中可以包含来自组织的不同部门的服务,甚至还能包含来自组织外的服务。如果没有恰当的控制,这种系统很容易失控。控制对所有生命周期阶段起到巩固支撑作用,为整个SOA系统提供指导,并有助于了解系统全貌。它提供指导和控制,帮助服务提供者和使用者避免遇到意外情况。

#### 5. 企业服务总线ESB

企业服务总线ESB(Enterprise Service Bus)是SOA基础架构的关键组件,是SOA架构的一个支柱技术。作为一种消息代理架构它提供消息队列系统,使用诸如SOAP或JMS(Java Message Service)等标准技术来实现。有人把ESB描述成一种开放的、基于标准的消息机制,通过简单的标准适配器和接口来完成粗粒度应用(比如服务)和其他组件之间的互操作。

在SOA体系架构中,企业服务总线ESB提供了服务管理的方法和在分布式异构环境中进行服务交互的功能,通过ESB,实现服务的部署、配置、注册、消息处理、消息路由、交互、事件侦听、执行、服务质量和服务级别管理等。ESB由中间件技术实现并作为支持SOA的一组基础架构,支持异构环境中的服务、消息,以及基于事件的交互,并且具有适当的服务级别和可管理性。

#### 6. SOA和Web服务的关系

Web服务与SOA有着很多相同的技术特点,如:基于XML语言,符合SOAP、WSDL和UDDI标准等。但是,SOA不同于Web服务。SOA是一种设计原则,是一个概念,是软件架构的方法学;Web服务则属于技术规范,是一种具体的实现技术。Web服务可以用来实现SOA,但是如果没有Web服务,企业照样也可以很好地实现SOA。

## 1.5 Web发展趋势

Web,这个由无以计数的超链接形成的网络世界,在给我们每一个人的工作、学习、生活和娱乐带来了无穷的便利和全新的生命体验的同时,新的技术、新的理念不断出现,又让我们对Web的未来充满了无限的遐想。

### 1.5.1 Web 2.0

回想Web诞生之初,我们面对一个个静态的网页,但网页之间的超链接,以及浏览器带给我们的网络界面,已经足够令我们兴奋不已。今天人们习惯地把这个时期(2003年以前

的互联网模式)的互联网称为 Web 1.0,这是一个信息消费的时代,人们通过浏览器获取信息。在 Web 1.0 时代,Netscape 脱颖而出,成为互联网耀眼的新星,它的浏览器把我们广大的普通用户带入了互联网。同时,Yahoo 提出了互联网黄页,Google 推出了深受欢迎的搜索服务,他们为互联网的发展做出了巨大的贡献。

### 1. Web 2.0 的概念

随着网络的发展,网站的拥有者发现,只有网民的参与,才能持久地提高与保持网站的人气。从一开始出现的"论坛"到快速火热起来的"博客",互联网事实上已经逐渐开始了一种理念上的转变,实践着从 Web 1.0 到 Web 2.0 的跨越。

Web 2.0 是 2004 年 3 月奥莱理(O'Reilly)媒体公司与 MediaLive 公司的一次头脑风暴会议上由 O'Reilly 媒体公司负责在线出版及研究的副总裁戴尔・多尔蒂和 MediaLive 公司的克瑞格・克莱共同提出。2005 年 9 月 30 日,O'Reilly 媒体公司主席兼 CEO 提姆・奥莱理在其公司网站的个人栏目中发表文章"什么是 Web 2.0——下一代软件设计模式和商业模式",成为 Web 2.0 理念提出的一个重要里程碑。

关于 Web 2.0,并没有一个统一的定义。互联网协会对 Web 2.0(互联网 2.0)的定义是:Web 2.0 是互联网的一次理念和思想体系的升级换代,由原来的自上而下的由少数资源控制者集中控制主导的互联网体系转变为自下而上的由广大用户集体智慧和力量主导的互联网体系。Web 2.0 内在的动力来源是将互联网的主导权交还个人,从而充分发掘了个人的积极性,使个人参与到体系中来,广大个人所贡献的影响与智慧和个人联系形成的社群的影响就替代了原来少数人所控制和制造的影响,从而极大解放了个人的创作和贡献的潜能,使得互联网的创造力上升到了新的量级。

如果说 Web 1.0 的主要特点在于用户通过浏览器获取信息,那么 Web 2.0 则更注重用户的交互作用,用户既是网站内容的消费者(浏览者),也是网站内容的制造者。可见,把 Web 2.0 理解成一个历史学的概念比技术性的概念更加准确,对这个概念的梳理,能帮助我们更好地把握互联网正在发生的技术与文化的变化。

### 2. Web 2.0 网站的特征

Web 2.0 是一种网站构建的理念,主要体现为以下特征:

(1) 网站平台应该有用户自己来定义的信息分类/分众标签功能和提供信息聚合 RSS (Rich Site Summary,或 Really Simple Syndication)功能,RSS 是一种描述和同步网站内容的格式,是目前使用最广泛的 XML 应用。

(2) 网站内容应该主要来源于用户原创内容(User Generated Content,UGC),即此网站的内容和信息更多地在于用户参与和用户主动的提供而不是通过大量转载和连接,有少数的内容是由网站编辑人员来控制和发布的。

(3) 网站管理者对其内容很少进行强制性的发布,而是在平台搭建完成后,对 UGC 的内容信息进行引导并促使其发生转变。

(4) 网站的推广和传播模式以用户间的相互传播为主,其产品使用过程中,鼓励使用者进行用户群体之间的传播。

(5) 网站的主要赢利模式是基于以上 UGC 内容及信息大量聚合后所释放的能量而进

行的，无论是在线的广告业务、ISP 增值业务还是其他收费模式，都需要在这个平台上发生。

(6) 注重用户体验的持续服务，服务和应用无处不在，不仅仅是少数重要用户，而是渗透到全体用户，包括大量的普通用户，要有拉动长尾的能力。

Web 2.0 理念使得网站的展现形式更加多样化，产生了很多的典型产品，例如：论坛、名人博客等。其实，无论是 Web 1.0 还是 Web 2.0，用户的体验是最重要的。随着互联网应用的更加普及，人们不仅是信息的消费者，越来越多的人会为网络提供信息，成为信息的制造者和提供者，并且因为参与了网络信息的提供而感到充实和自我实现。

## 1.5.2　语义 Web

万维网已经成为人类最大的信息仓库，而且各种语言、各个知识领域的内容还在源源不断地快速增长。这些海量的信息在给我们提供便利的同时，让我们的信息查询变得极为困难。大多数的搜索引擎是基于关键词的搜索，搜索是基于页面内容的，而不是基于页面信息的语义，查准率较低。人们需要让页面内容有意义，从而提供各种依靠语义的自动化服务，这就是语义 Web 的研究动机。

### 1. 语义 Web 的概念

在 WWW 出现不久，人们就已经意识到语义对于 Web 的重要性。HTML 只是规范了信息的显示，却无法表达内容的含义。没有形式化的网页内容，机器将无法实现信息处理的自动化。只有将网页内容表述成机器可以理解的格式，Web 才可能成为一个巨大的知识库，充分实现信息的查找、共享和重用。为此，1998 年，万维网的发明者 Tim Berners-Lee 首次提出了语义 Web(Web Semantic)的概念。

对于语义 Web 的概念，一般表述是：语义 Web 是当前 Web 的一个扩展，其中信息具有形式化定义的语义，更有助于计算机之间以及计算机与人之间的协同工作。其思想是使 Web 上的数据以这样一种方式来定义与链接，使其能够在各种不同的应用场景中有效地实现数据的发现、自动化处理、集成与复用。当且仅当 Web 不仅成为人所共享加工的场所，也成为自动化工具所共享加工的场所时，语义 Web 方能实现其全部潜力。

语义 Web 有很多突出的优点，包括数据集成更简单、搜索更精确、知识管理更方便，等等，结果语义 Web 这个词的含义越来越丰富。

### 2. 语义 Web 体系架构

要实现语义 Web，依赖于三大关键技术：XML、RDF 和 Ontology。语义 Web 分层体系架构如图 1-5 所示。

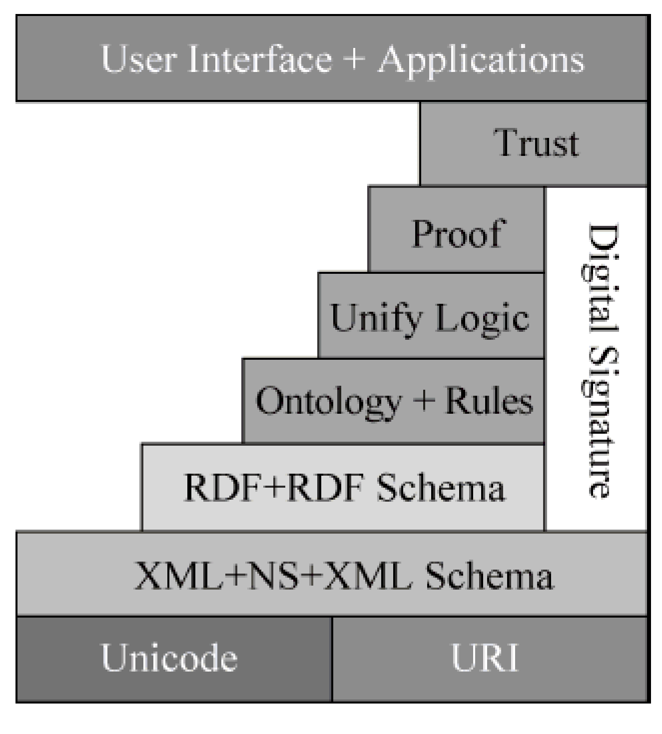


图 1-5　语义 Web 体系架构

(1) Unicode 和 URI 层：Web 内容采用 Unicode 字符集，负责资源的编码，统一资源定位符 URI(Uniform Resource Identifier)用于资源标识，唯一标识网络上的一个概念或资源。在语义 Web 体系结构中，该层是整个语义 Web 的基础。

(2) XML＋NS＋XML Schema 层：该层可以称为

XML 层,用于表示数据的内容和结构。XML 实现文档的结构化定义,即进行文档形式化。命名空间 NS(Name Space)由 URI 索引确定,目的是防止元素重命名而引起的歧义。XML Schema 提供更多的数据类型,为 XML 文档提供数据校验机制。该层负责从语法上表示数据的内容和结构,实现 Web 内容和表现形式的分离。

(3) RDF+RDF Schema 层:又可以分为 RDF 层和 RDF Schema 层,其中 RDF 用于描述资源及其相互关系,RDF Schema 层为 RDF 提供类型定义机制,确定 RDF 描述的资源所使用的领域词汇。因为 XML 不具备语义描述能力,W3C 推荐以 RDF(Resource Description Framework)标准来解决 XML 的语义局限。

RDF 解决的是如何采用 XML 标准语法无二义性地描述资源对象的问题,使得所描述的资源的元数据信息成为机器可理解的信息。RDF Schema 使用一种机器可以理解的体系来定义描述资源的词汇,其目的是提供词汇嵌入的机制或框架,在该框架下多种词汇可以集成在一起实现对 Web 资源的描述。

(4) Ontology+Rules 层:本体(Ontology)负责在 RDF(S)基础上定义的概念及其关系的抽象描述,用于描述应用领域的知识,描述各类资源及资源之间的关系,实现对词汇表的扩展。在这一层,用户不仅可以定义概念而且可以定义概念之间的复杂关系。规则用于描述领域知识中的前提和结论。本体和规则共同构成领域知识层。

W3C 推荐使用 OWL(Web Ontology Language)作为 Web 本体描述语言,OWL 既提供了正式的语义,又提供了附加的词汇,比起 XML、RDF 和 RDF Schema,对 Web 内容实现了更好的机器互操作性。

(5) Unify Logic 层:该层负责在下面各层基础上,提供公理和推理规则,而 Logic 一旦建立,便可以通过逻辑推理对资源、资源之间的关系以及推理结果进行验证,证明其有效性。

(6) Proof 层:通过 Proof 交换以及数字签名,建立一定的信任关系,从而证明语义 Web 输出的可靠性以及其是否符合用户的要求。

(7) Trust 层:支持代理间通信的证据交换,在用户间建立信任关系。

(8) User Interface+Applications:应用层是构建在语义 Web 之上的各种应用。

总之,在语义 Web 体系架构中,下面两层是语义 Web 的基础设施,中间从元数据发展到本体描述语言及其统一的逻辑是语义 Web 的关键,证明和信任及各层次贯穿的数字签名技术是扩充,是对语义 Web 成功应用的要求与展望。

### 3. 语义 Web 的应用

在语义 Web 中,可以提供各种依靠语义的自动化服务,包括:(1)互联网信息发布与搜索,通过对内容的标注与分析从而克服关键词查询的歧义性,可以大大提高查询精度。此外,基于语义 Web 的文档检索与知识管理也是当前研究的一个热点。(2)Web 问题解答,在用 Ontology 对信息源进行标注的基础上,进一步运用知识库来解答用户的提问。例如,Stanford 大学研制的 Triple 系统是一个基于逻辑程序设计的 RDF 查询系统,逻辑子句的问题求解能力使它能够解答较为复杂的问题。德国 Karlsruhe 大学等单位研制的 SEAL 是一个语义 Web 门户网站,它具有回答用 F-逻辑表示的查询的能力,F-逻辑使得 Ontology 中的概念与问题求解规则融合于一体。

语义 Web 的目标是改善当今的 Web,它的主要思想是使语义信息成为计算机可处理

的对象。要将 Web 语义化是非常困难的，语义 Web 很难一下子获得巨大成功，但是，它会一点点地渗透到现有的 Web 中，在人们的不知不觉中，让我们进入语义 Web 的时代。

## 思　考　题

1. 什么是万维网？
2. 什么是 Web 服务器？简述 Web 的基本工作机理。
3. 解释下列概念：
   网站、网页、HTTP、URL、端口、超文本
4. 什么是 B/S 结构？和 C/S 结构相比，有什么优点？
5. 简述你对 Java 技术的理解。
6. XML 技术标准体系是如何划分的？列举常用的 XML 基础标准，并说明它们的功能。
7. 什么是 Web 服务？Web 服务基于哪些主要的技术标准？
8. 简述软件开发模式的演化过程。
9. 什么是 SOA？简述 SOA 和 Web 服务的关系。
10. 什么是 Web 2.0？它有哪些主要特征？
11. 什么是语义 Web？画出语义 Web 的分层模型，并说明各层的功能。

# 第2章

# Web服务器的架设和管理

在 Internet 中,Web 服务是最主要的网络服务,几乎一提到 Internet,就会想到万维网(World Wide Web,WWW)。实际上,WWW 只是 Internet 的一个子集,它是由 Internet 中的 Web 服务器和 Web 客户机构成的。Web 服务器就是那些安装了 Web 服务器软件的计算机,而安装了浏览器的计算机就是 Web 客户机。在互联网中,每一个网站的背后都运行着一个 Web 服务器,利用 Web 服务器,不仅能够提供信息发布,更重要的是,它突破了局域网的限制,将计算机应用扩展到整个互联网中,彻底改变了网络环境下的计算模式。通过 Internet,企业可以部署 Web 应用,从而建立基于 B/S 结构的应用系统。

## 2.1 操作系统与 Web 服务器

在全球数亿个网站或 Web 应用系统的背后,都运行着一个 Web 服务器。要使一台计算机成为 Web 服务器,需要安装网络操作系统和相应的 Web 服务组件。不同的操作系统平台,安装的 Web 服务组件也不相同。Web 服务组件从本质上讲就是一个网站或 Web 应用的运行环境,因此,安装的 Web 服务组件不同,Web 应用的开发工具也不相同。

目前,主流的 Web 服务器软件主要由 IIS 或 Apache 组成。IIS 支持 ASP 且只能运行在 Windows 平台下,Apache 支持 PHP、CGI、JSP 且可运行于多种平台。虽然 Apache 是世界使用排名第一的 Web 服务器平台,但是,由于 Windows 的易用性,因此具有很大的装机数量,IIS 的应用也很多。

### 2.1.1 Web 服务器

在 Internet 中,Web 服务器有两个层面的含义,一是指安装了 Web 服务器的计算机,第二层含义是指 Web 服务器程序。所谓 Web 服务器程序,简单地讲就是一个服务程序,它仅仅需要监听合适的端口,建立连接,然后发送数据。服务器程序的开发总是和客户端软件的开发相辅相承的。

在 20 世纪 90 年代,Web 服务器、浏览器、HTML 和 HTTP 协议都得到了快速的发展,这标志着 WWW 的诞生。美国国家超级计算应用中心 NCSA 开发了 httpd 代码,实现了 Web 服务器的功能。同时,NCSA 开发了最早的浏览器,这就是 NCSA Mosaic 浏览器,它

成为后来 Netscape、微软 IE 以及众多网页浏览器的鼻祖。虽然 NCSA Web 服务器不再被维护和继续开发，但是仍然可以免费下载其源代码，它是著名的 Web 服务器 Apache 的前身。

随着 Internet 的发展，人们对 Web 服务器的功能提出了更多的需求。Web 服务成为很多商务应用都必须面对和采用的技术时，就出现了很多不同 Web 服务器以满足这些不同的需求。在电子商务应用环境中，可伸缩性、可靠性和高级动态功能都是 Web 服务器应该具有的关键因素。此外，易于配置和管理对于新手来说也同样重要。对于这些所有的特性来说并没有任何一个特定服务器能完全满足需求，但是如果对 Web 服务的需求有明确的认识，可以便于从不同的 Web 服务器中选择适合的服务器。

## 2.1.2 主流 Web 服务器简介

Web 服务器产品很多，同一种 Web 服务器往往也有不同的操作系统版本，下面对各种不同的 Web 服务器进行简要介绍。

### 1. Internet Information Server(IIS)

Internet Information Server(IIS)是 Windows 自带的服务器组件。在 Windows 2000 Server 中，自带了 IIS 5.0，Windows Server 2003 中自带了 IIS 6.0。现在，微软已经全部采用 Windows Server 2003/IIS 6.0，而不再安装 Windows 2000 Server/IIS 5.0 了。

在 IIS 中，包含了一系列的 ASP 对象，负责 ASP 页面中服务端脚本程序的解析工作，同时，为用户开发基于 Web 的应用提供一个开发环境。IIS 6.0 支持 Web 应用的.net 开发环境。

作为 Windows 平台的内置服务组件，其优点是容易安装和管理，但是最大的缺点是只能安装在 Windows 平台中，不能在其他的操作系统平台上安装。

在 Windows 服务器上，除了安装 IIS 外，还可以安装 Apache 和 Tomcat。其中，Tomcat 是 Serverlet/JSP 的容器，可以运行 JSP 脚本程序，开发基于 JSP 的 Web 应用系统。

### 2. Apache 服务器

Apache 源于 NCSA httpd 服务器，早期 Apache 的开发是通过对 NCSA 的 httpd 代码添加补丁程序来进行的，取名为“a patchy server”，即“补丁服务器”。Apache 被看做是“补丁服务器”，还因为它具有模块化特性，该特性实现了 Apache 的灵活性和可扩展性，而且开发者可以利用该特性很容易地添加第三方功能模块，通过模块，开发人员可以添加任何功能。

在发展初期，Apache 主要是一个基于 UNIX 系统的服务器，它的宗旨就是建成一个基于 UNIX 系统的、功能更强、效率更高并且速度更快的 WWW 服务器。目前，Apache 是使用最广的 Web 服务器，有多个操作系统平台版本，可以运行在几乎所有的 UNIX、Windows、Linux 等主流操作系统平台上，并且很多类型的 UNIX 操作系统都集成了 Apache。

因为 Apache 服务器具有简单、高效、性能稳定、安全、免费等特性，已经成为最为广泛的 Web 服务器。许多大型的网站，例如：Google、Yahoo、阿里巴巴、Sina、百度、网易、搜狐等都采用 Linux 或 FreeBSD 等操作系统平台，并配置 Apache 服务器，构建自己的 Web 服务器。在版本上，大多数公司应用 Apache 2.0 或 Apache 2.2.x。

### 3. 其他 Web 服务器

除了 Apache 和 IIS 外,还有许多不同特点的 Web 服务器,这些服务器包括:

(1) Zeus Webserver 服务器:Zeus 是一个商业化的 Web 服务器产品,在 SMP 环境下有优秀的可伸缩性,并实现了常见的特性集合,如访问控制、动态内容产生和安全等。具有健壮、集成有集群支持的容错和负载平衡等特色,是高端应用的很好的选择。

(2) iPlanet 服务器:iPlanet 是 Sun、Netscape 和 AOL 公司联合生产的 Web 服务器产品,iPlanet 和其他 Netscape 产品一样具有很高的性能,而且 iPlanet 具有 Sun 公司 Java 的特性。iPlanet 具有现今高性能 Web 服务器的特性,相对于其他 Web 服务器,iPlanet 能够提供更多的 Java 功能,能够运行标准 Java API,并且在 Java API 环境下运行速度良好。

(3) AOLserver Web 服务器:AOLserver 的研究始于 1994 年,当时它作为 Web 发布系统的一部分进行开发。在该 Web 发布系统中内嵌了 Web 服务器的 WYSIWYG 网页编辑器,该网页编辑器强调内容变化的便利性和内容更新的快捷性。早期的 Web 发布系统被设计成一个完整的网页编辑系统。随着时间的推移,AOL 公司的网页编辑器已经不复存在,但是由于 Tcl 脚本语言的出现和它对动态网页的支持,AOLserver 却生存了下来。

Web 服务器产品很多,随着互联网技术的发展,新的 Web 服务器产品还将不断出现。选择 Web 服务器时,服务器对动态脚本语言、API 的支持以及数据库连接的性能都是重要的因素。此外,Web 服务器的配置和管理,也是选择一个 Web 服务器的重要因素。

## 2.2 使用 Internet 信息服务

在 Windows 操作系统平台上,内置了 Internet 信息服务(Internet Information Server,IIS)组件,用于创建 Web 服务器。作为 Window 平台,最常用的是 Windows 2000 Server 和 Windows Server 2003,其中 Windows 2000 Server 内置了 IIS 5.0,Windows Server 2003 内置了 IIS 6.0。目前,Windows Server 2003/IIS 6.0 正逐步取代 Windows 2000 Server/IIS 5.0,成为最主要的 Windows 平台 Web 服务器配置。

### 2.2.1 什么是 Internet 信息服务

Internet 信息服务(Internet Information Server,IIS)是一组 Windows 操作系统组件,此组件可以使公司很方便地创建自己的 Web 服务器、FTP 服务器、E-mail 服务器和 NNTP 服务器,从而将信息和业务应用程序发布到 Web 中。在 IIS 内部,本质上是由一系列的 ASP 对象组成的,负责 ASP 页面中服务端脚本程序的解析工作,同时,为用户开发基于 Web 的应用提供一个开发环境。

在 Windows 2000 中,内置了 IIS 5.0。从 Windows Server 2003 开始,IIS 升级为 IIS 6.0,将 IIS 5.0 中的 SMTP 服务器升级为完整的 E-mail 服务器。IIS 由若干可选组件构成,用户可以根据需要选择不同的组件进行安装和配置,IIS 包含的组件如下所述。

#### 1. Internet 服务管理器

用于配置和管理 IIS，可以在 MMC 中以管理单元形式显示，该管理工具还在控制面板的“管理工具”文件夹中创建一个快捷方式。

#### 2. Internet 服务管理器(HTML)

基于 HTML 的 Internet 服务管理器，可以使用浏览器对 IIS 进行远程管理。

#### 3. NNTP Service

NNTP(Network News Transfer Protocol)，即网络新闻传输协议，是 TCP/IP 协议套件的成员。负责将新闻函件分发到 Internet 上的 NNTP 服务器和 NNTP 客户端，设置了 NNTP 后，就可以将新闻文章存储在服务器上的中央数据库，用户可以选择指定的项目阅读。

#### 4. SMTP Service

SMTP(Simple Mail Transfer Protocol)，即简单邮件传输协议，是 TCP/IP 协议套件的成员，用来管理邮件代理之间的电子邮件交换。

#### 5. World Wide Web 服务

Web 服务，用于对 Web 站点的创建、管理以及为用户访问 Web 服务器提供服务，内置服务端脚本引擎，是 ASP 等服务器脚本规范的容器。

#### 6. 文档传输协议 FTP 服务器

用于建立 FTP 站点，支持文件的上传和下载。

### 2.2.2　安装 IIS

在 Windows 2000 Server 下，安装 Windows 2000 Server 时，默认情况下，IIS 5.0 被一并安装。在 Windows Server 2003 中，IIS 组件是“应用服务器”的一部分，可以在安装操作系统时选择安装，也可以通过“添加/删除 Windows 组件”方式来安装 IIS，或者通过管理工具中的“管理您的服务器”程序添加“应用程序服务器”角色，来完成 IIS 6.0 的安装。

下面以 Windows Server 2003 企业版为例，说明 IIS 的安装，具体操作步骤如下：

(1) 将 Windows 2003 Server 系统光盘插入光盘驱动器。

(2) 在“控制面板”窗口中，双击“添加/删除程序”图标，在“添加/删除程序”窗口中，单击“添加/删除 Windows 组件”，打开“Windows 组件向导”对话框，在组件列表中，选择“应用程序服务器”(在 Windows 2000 Server 中为 Internet 信息服务)，然后单击“详细信息”按钮，显示“应用程序服务器”对话框，如图 2-1 所示。

在应用程序服务器组件列表中，选择“Internet 信息服务”，然后单击“详细信息”，打开“Internet 信息服务子组件”对话框，显示 Windows Server 2003 中相关组件，如图 2-2 所示。

单击“万维网服务”，然后单击“详细信息”按钮，打开“万维网服务”对话框，显示万维网服务子组件列表，如图 2-3 所示。

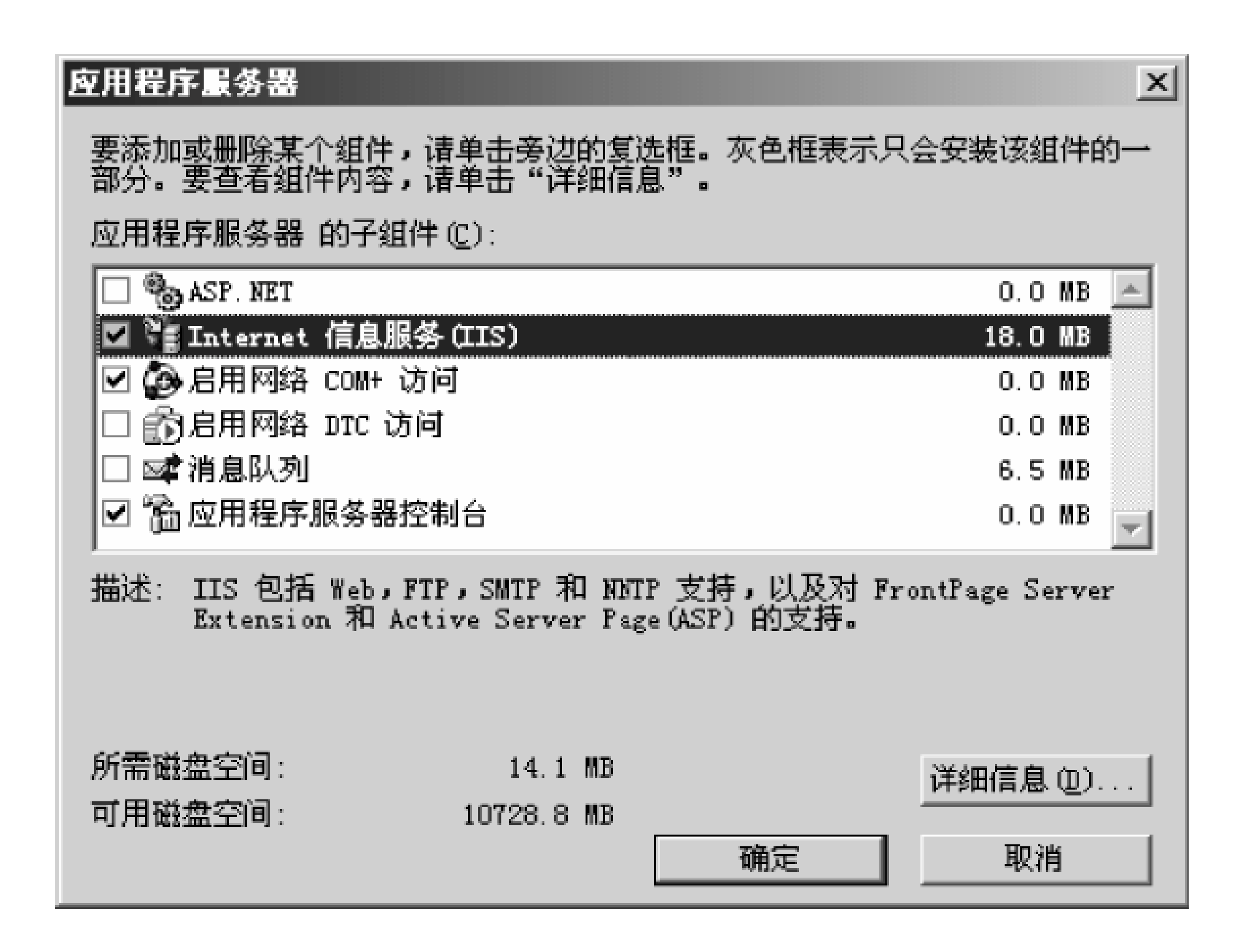



图 2-1　Windows Server 2003 的应用程序服务器组件列表

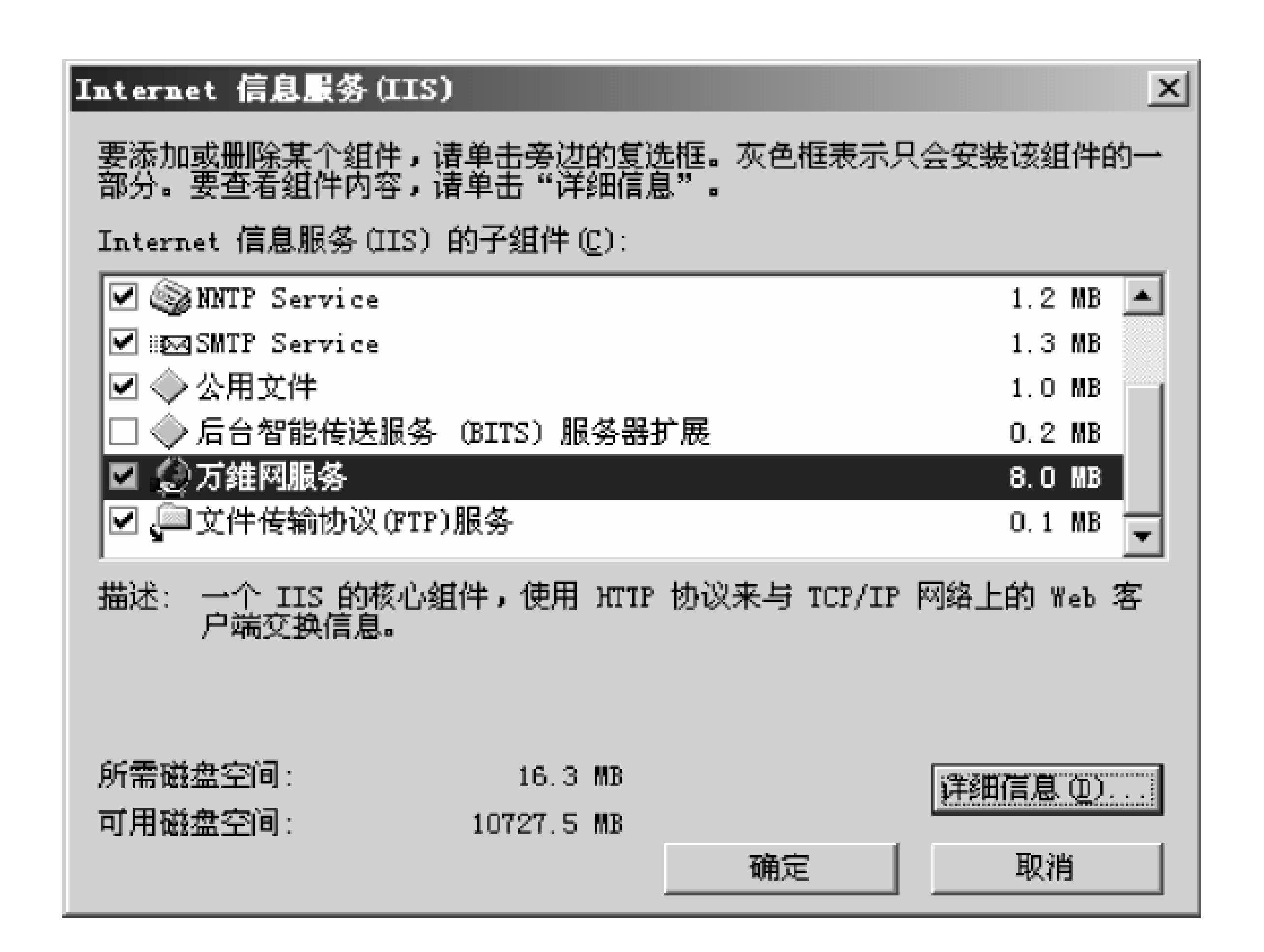



图 2-2　Internet 信息服务(IIS)组件列表

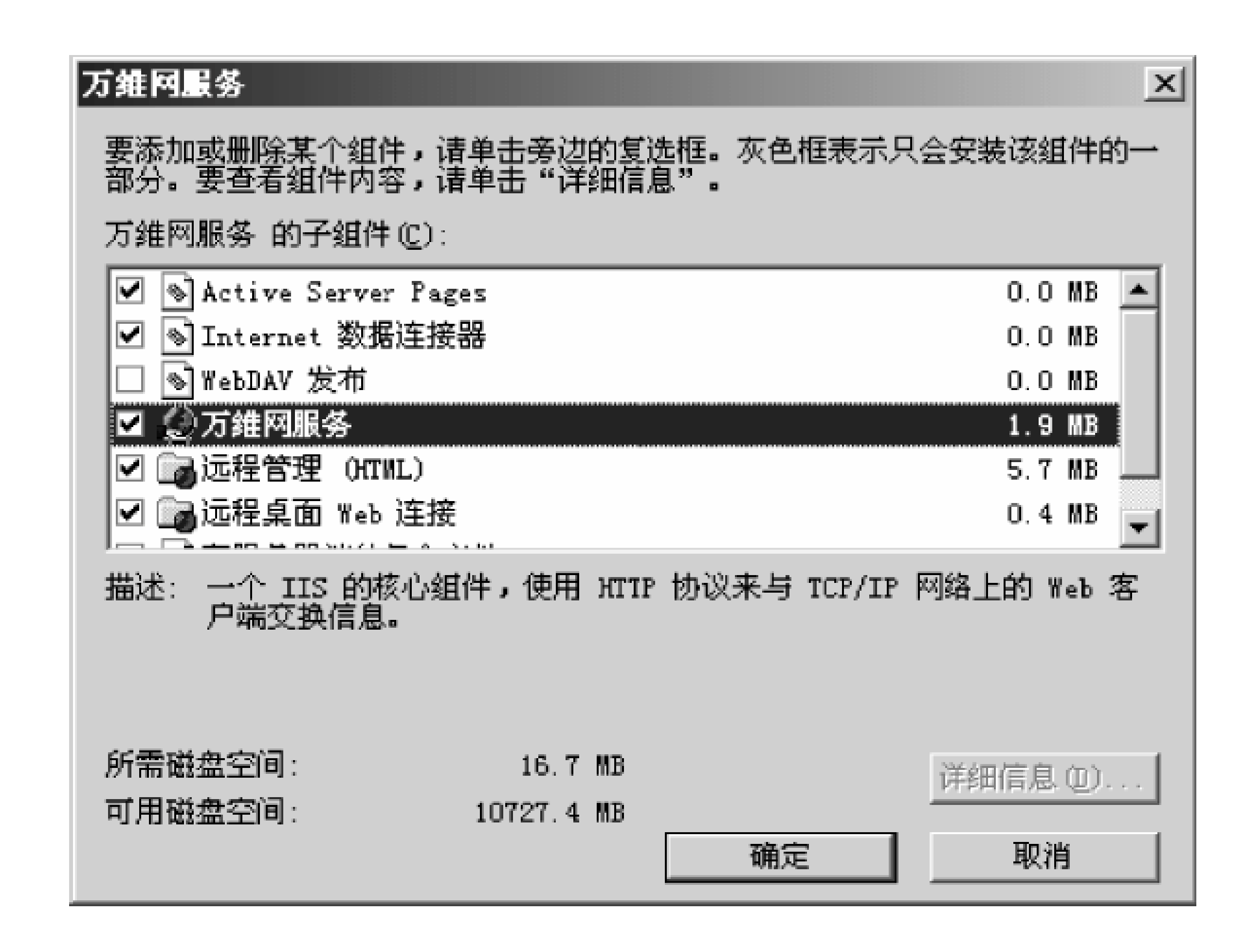



图 2-3　Internet 信息服务组件列表

在万维网服务子组件列表中，选择相应的组件，包括：万维网服务、远程管理和远程桌面 Web 连接，然后单击“确定”按钮，向导从光盘复制文件并进行相关的配置。安装结束后，在“控制面板”的“管理工具”中将增加“Internet 信息服务(IIS)管理器”、“远程桌面”等程序。同时，在服务器 C 盘根目录下将创建一个 Inetpub 文件夹，在该文件夹下创建多个子文件夹，文件夹结构如图 2-4 所示。




图 2-4　安装 IIS 6.0 后自动创建的相关文件夹

各文件夹说明如下：

- AdminScripts 文件夹：存储 CGI 脚本的根目录。
- ftproot 文件夹：FTP 服务根目录。
- mailroot 文件夹：SMTP 服务器根目录。
- nntpfile 文件夹：新闻组信息根目录。
- wwwroot 文件夹：默认 Web 站点的根目录。

## 2.2.3 Internet 信息服务管理器

IIS 安装完成后，在 Web 服务器的“管理工具”文件夹中增加“Internet 服务管理器”工具。同时在“计算机管理”控制台中，在“服务和应用程序”结点下增加“Internet 信息服务”结点。通过 Internet 服务管理器，可以创建 Web 站点、FTP 站点，以及对它们进行配置和管理。对 IIS 的管理可以有多种途径，具体介绍如下。

### 1. Internet 信息服务管理器

单击“开始”按钮，选择“程序”、“管理工具”，执行“Internet 信息服务管理器”命令可以直接启动“Internet 信息服务管理器”，如图 2-5 所示。




图 2-5 "Internet 信息服务管理器"控制台

### 2. Internet 信息服务管理单元

安装了 IIS 后,"Internet 服务管理器"作为一个管理单元,被组织到"计算机管理"控制台中。在"控制面板"、"管理工具"中,双击"计算机管理"打开"计算机管理"控制台,可以显示"Internet 信息服务(IIS)管理器"单元,如图 2-6 所示。

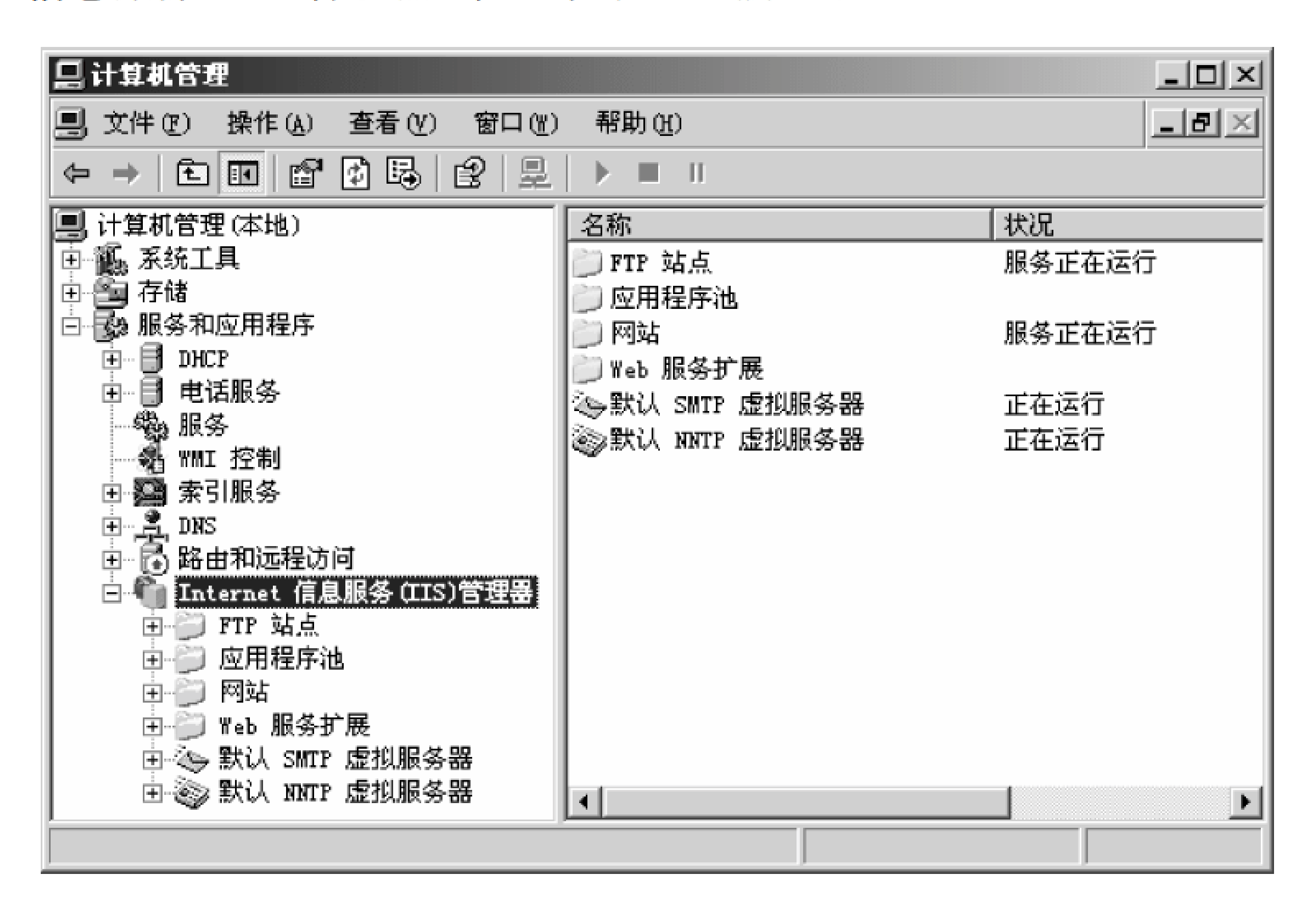



图 2-6 "计算机管理"控制台中的"Internet 信息服务管理器"单元

## 2.3 创建 Web 站点

在 Windows 平台中,当安装了 Internet 信息服务(IIS)后,就可以创建 Web 站点了,从而把该计算机配置为 Internet 中的一台 Web 服务器,向用户提供 Web 连接服务。

### 2.3.1　创建 Web 站点

在 Windows 服务器/IIS 中，可以利用 IIS 创建和管理 Web 站点。下面以 Windows Server 2003 企业版为例，介绍 IIS 中 Web 站点的创建过程。

单击“开始”按钮，选择“程序”、“管理工具”，单击“Internet 服务(IIS)管理器”，打开“Internet 信息服务”控制台，右击“网站”结点，打开快捷菜单，如图 2-7 所示。




图 2-7　新建 Web 站点

在快捷菜单中选择“新建”，单击“网站”命令，启动“网站创建向导”，然后，单击“下一步”按钮，如图 2-8 所示。




图 2-8　输入网站描述

在网络描述界面，输入 Web 站点的说明(即新站点的名称)，该名称将在“Internet 信息服务(IIS)管理器”控制台中显示。单击“下一步”按钮，如图 2-9 所示。

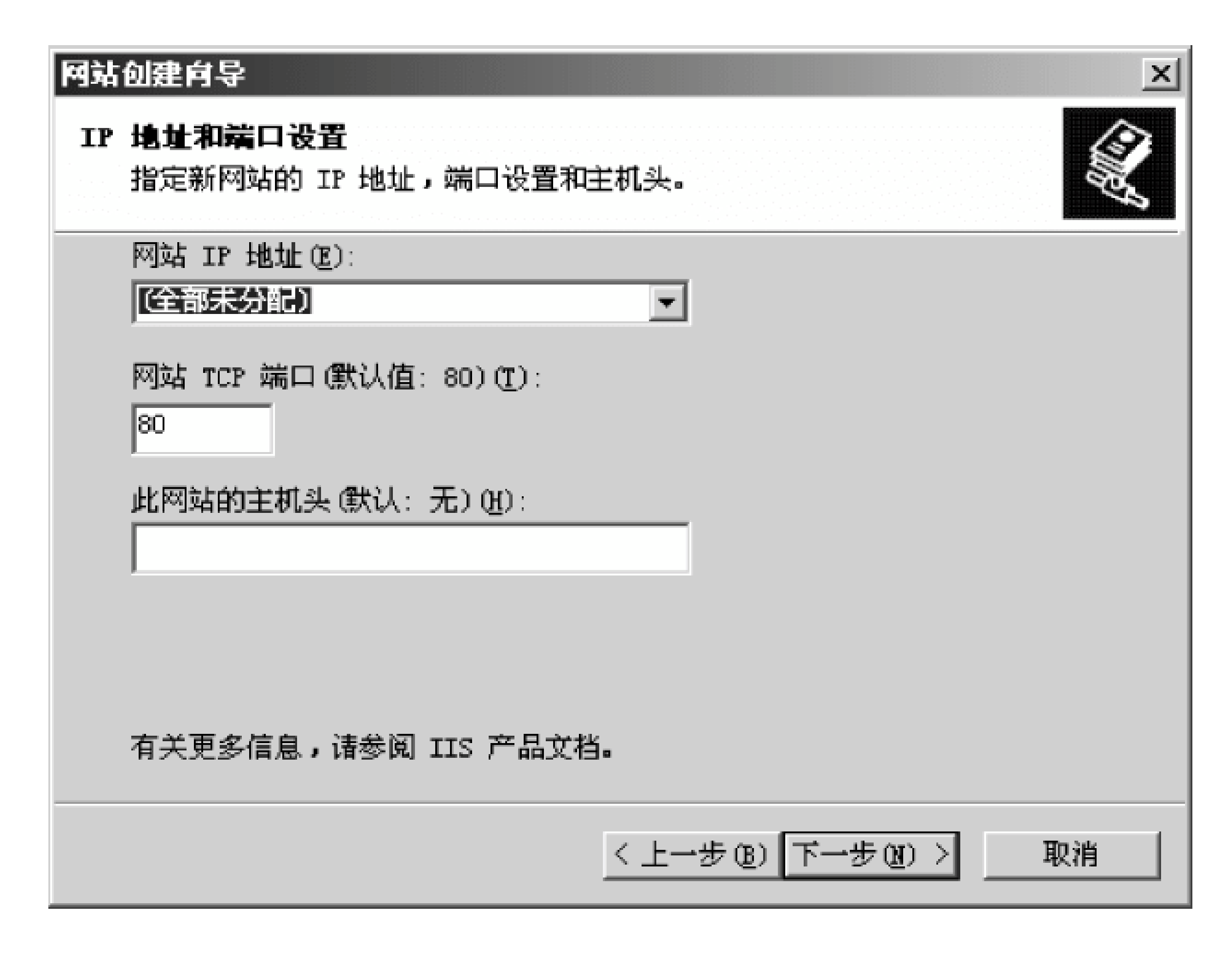



图 2-9　设置网站 IP 地址和端口

在 IP 地址后面的下拉列表中，默认显示“全部未分配”，单击下拉列表，可以显示网卡设置的多个 IP 地址。可以选择“全部未分配”或从中选择一个 IP 地址。

一般情况下，一个 Web 站点有一个 IP 地址。如果一台机器上只有一个 IP 地址，又需要运行多个 Web 站点，此时可以为不同的 Web 站点指定不同的端口号。指定不同的端口号后，要连接到该站点，在网址(IP 地址或域名)后需要给定对应的端口号，如：http://202.194.73.118：8080。使用非默认的端口号将使得客户端连接 Web 站点时，必须知道该站点的端口号，并且在 URL 中不能省略协议前缀 http://，否则使用不方便。

如果希望使用相同的 IP 地址，又保留 HTTP 默认的端口号 80，还可以使用不同的主机头来区分不同的 Web 站点。这里全部选用默认值，单击“下一步”按钮，如图 2-10 所示。

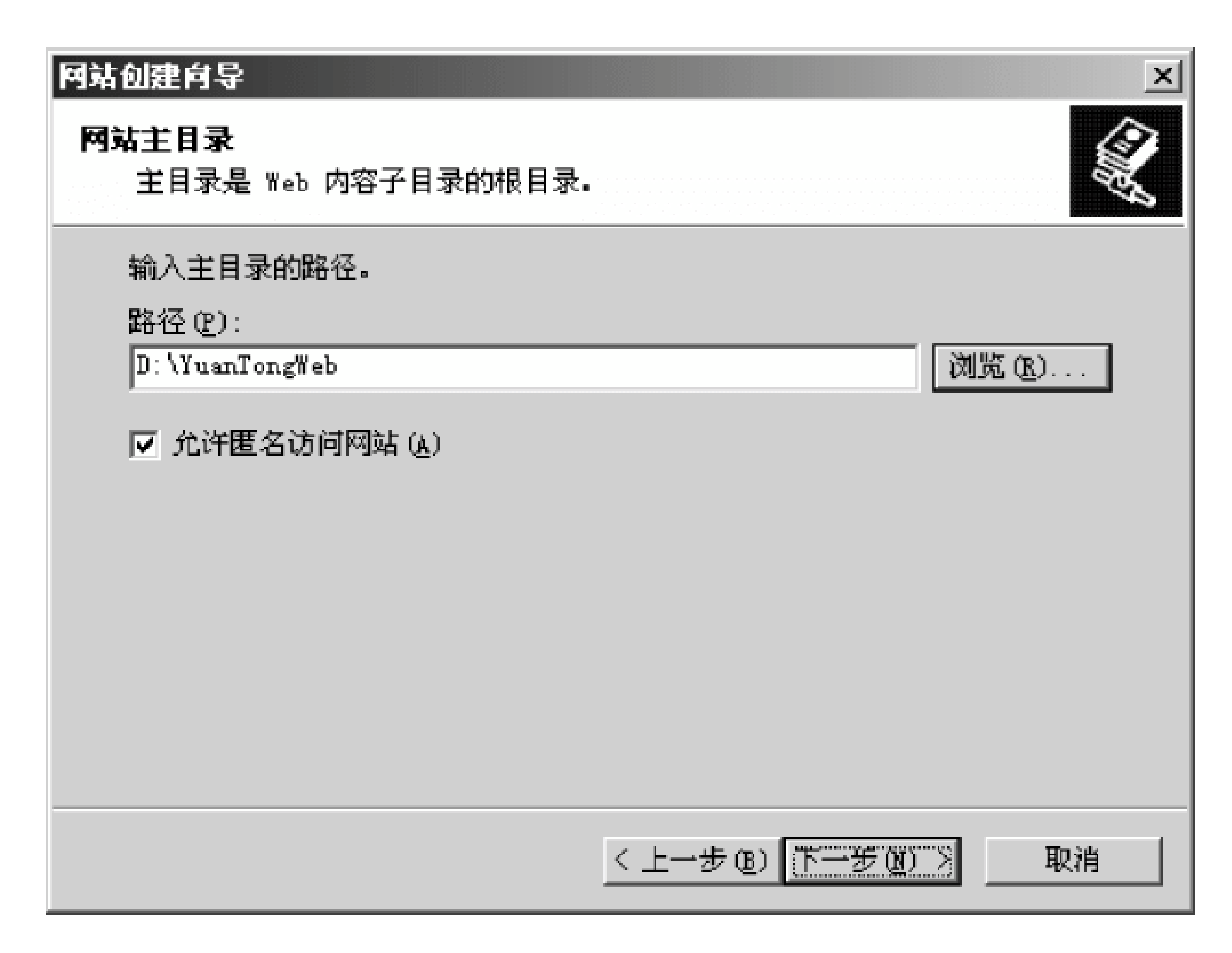



图 2-10　设置网站主目录

在路径下面的文本框中，可以输入该站点的主目录，或者通过“浏览”按钮选择一个目录作为网站主目录。主目录保存了一个 Web 站点中的所有内容，包括各个子文件夹以及所有的网页文件。站点主目录又称为站点的根目录，站点首页文件通常存储在站点的主目录下。

单击“下一步”按钮，显示网站访问权限设置界面，如图 2-11 所示。

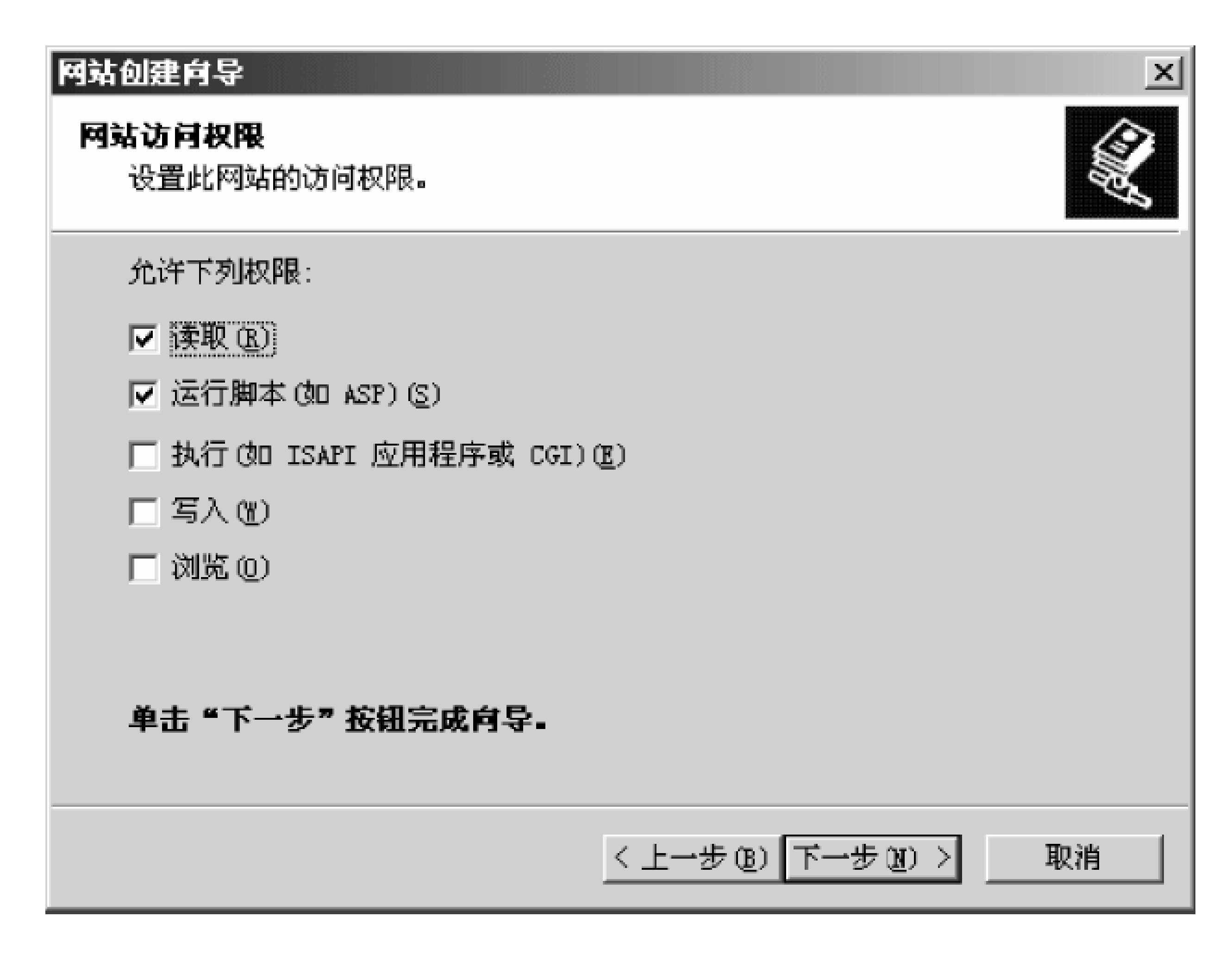



图 2-11 设置网站访问权限

根据需要，对 Web 站点的权限进行设置，从允许的权限中选择相应的权限。一般情况下，需要选择“读取”权限和“运行脚本”权限。有关权限设置的详细介绍请参见第 2.4 节“Web 站点的配置”。

然后单击“下一步”按钮，显示“已经成功完成 Web 站点创建向导”。最后单击“完成”按钮，返回到“Internet 信息服务管理器”控制台，如图 2-12 所示。

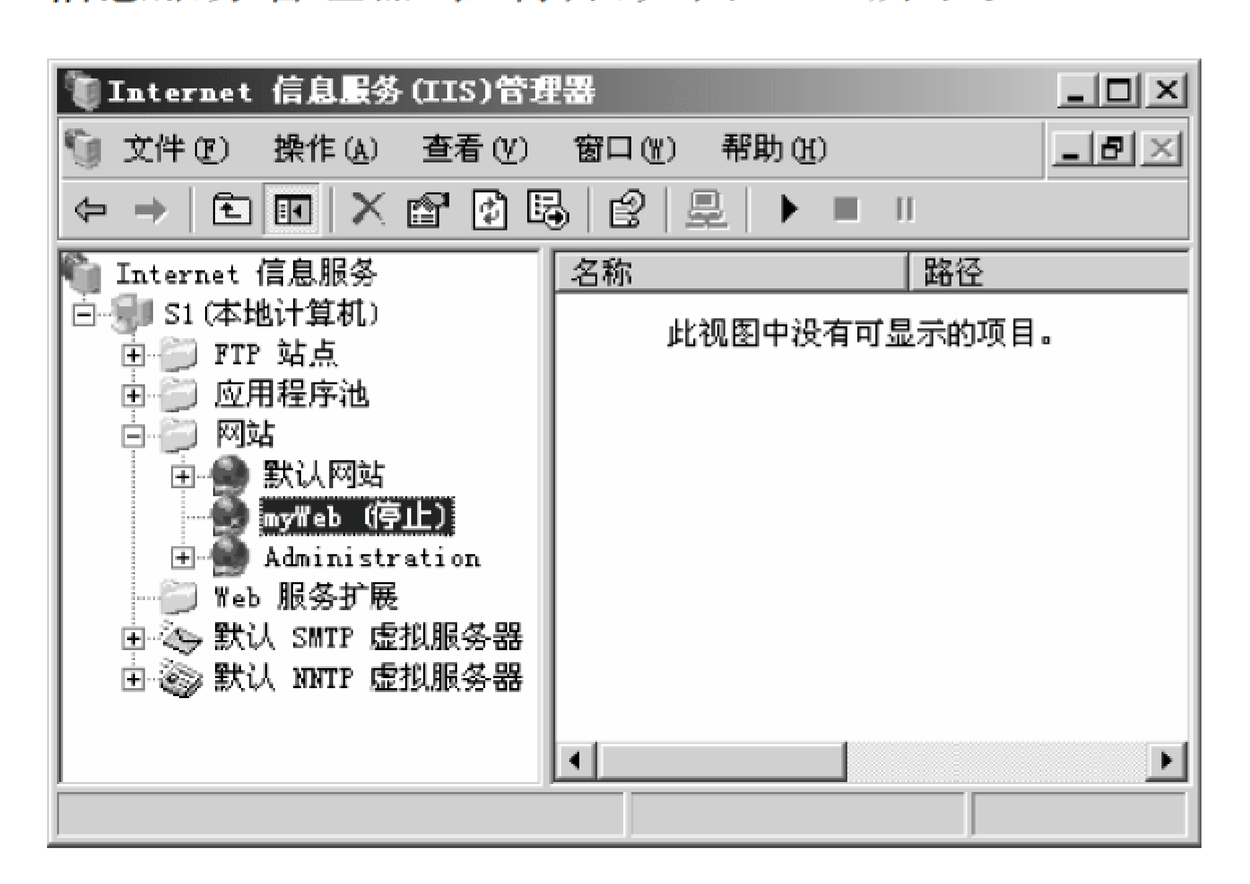



图 2-12 “Internet 信息服务管理器”控制台

新站点创建完成后，主目录中没有任何内容。如果新建的 Web 站点和已经存在的 Web 站点的 IP 地址和端口号完全一样，新站点将被标记为“停止”。

### 2.3.2 Web站点的启动、停止和暂停

上面我们看到由于新建站点和默认 Web 站点 IP 地址和端口号完全一样,使得新建站点被停止。如果要将停止的 Web 站点启动,右击被停止的 Web 站点,在快捷菜单中,选择“启动”命令,该站点将被启动。

如果要停止一个 Web 站点,右击该站点,在快捷菜单中,单击“停止”命令,该站点将被停止。

当管理人员需要维护系统或网页数据时,可以暂停 Web 站点,站点暂停后,它将不接受客户浏览器的连接,等用户工作结束后,再启动该站点。

如果用户试图连接一个暂停的站点,客户端浏览器显示“找不到该页”消息(HTTP 404-未找到文件)。如果试图连接一个停止的站点,客户端浏览器显示“该页无法显示”的消息(找不到服务器或 DNS 错误)。

### 2.3.3 规划 Web 应用

一个 Web 站点建立后,就意味着一个 Web 应用的开始。所谓 Web 应用,是指在 Internet 环境中应用程序新的开发和使用模式,它是 B/S 结构下应用程序的实现形式。一个 Web 网站可以简单地看做是一个 Web 应用,它由主目录、子目录及其包含的网页文件、图片文件及其他各类文件,以及相关的数据库构成。

#### 1. 网站首页

传统的应用程序都有一个主用户界面,包含菜单栏、工具条等,用户通过菜单命令或工具按钮执行特定的程序功能。在 B/S 结构中,一个 Web 应用则是从网站首页开始的,相当于传统的应用程序主用户界面。

首页(Home Page)是当客户连接到一个站点时首先看到的 Web 页面。在设计 Web 站点的首页时,不仅要考虑页面的外观、栏目布局,更重要的是在页面内容上必须包含可以到各种功能页面的超链接。首页的默认文件名一般为 index. htm、default. htm 等,首页文件通常需要保存在 Web 站点的主目录下。

#### 2. 规划网站的文件结构

一个网站,即一个 Web 应用,应该根据用户需求来设计网站的功能或栏目。为了管理方便,应该根据网站功能对网站文件夹结构进行认真规划。一般情况下,在主目录下往往需要创建多个子文件夹,每个文件夹对应网站的一个功能,存储相关的网页文件。对于一些公用的程序或图片,可以定义单独的文件夹。此外,还可以规划数据库文件夹,存储网站用到的数据库文件,便于整个网站的备份。

对于刚刚新建的网站,假设该网站设计有四个主要功能栏目:即时消息、在线聊天、bbs、个人博客,在主目录下可以分别创建四个文件夹,分别存储开发即时消息、在线聊天、bbs 和个人博客所用到的网页文件,文件结构可以如图 2-13 所示。




图 2-13　站点主目录下的内容组织

### 3. 使用虚拟目录

在一个网站中，网站主目录及其中的子文件夹，称为物理目录。逻辑上讲，只有主目录下的文件才是网站的组成部分。如果要把本机上其他文件夹，甚至是网络中其他计算机上的文件夹作为 Web 站点的内容，则需要使用虚拟目录。虚拟目录可以看做是 Web 站点主目录下指向其他物理目录的指针。

1）使用虚拟目录的好处

使用虚拟目录可以将 Web 站点的数据保存到本机上主目录以外的物理目录中，甚至是其他的计算机中，避免 Web 站点数据占用服务器太多的空间。

另外，当数据移动到其他的地址时，不会影响 Web 站点结构。此时不需要更改虚拟目录的名称，只需要重设虚拟目录，将虚拟目录指向新的物理目录即可。

2）建立虚拟目录

要建立虚拟目录，可按照下面的步骤操作：在“Internet 信息服务”控制台目录树中，右击某 Web 站点，在快捷菜单中，指向“新建”，单击“虚拟目录”，启动“虚拟目录创建向导”，按照向导提示操作，依次输入“虚拟目录别名”、虚拟目录对应的实际物理目录，以及设置虚拟目录的访问权限。最后单击“完成”按钮，返回“Internet 信息服务(IIS)管理器”控制台，即可显示新建的虚拟目录。

使用虚拟目录，可以实现在一个 Web 应用中简单地增加其他页面，而这些页面和现有的内容可能是没有直接关系的，它可能是临时性的。此时只需要通过 http://域名或 IP 地址/虚拟目录/文件名(包括扩展名)访问主目录以外的文件。其中，http://域名或 IP 地址/代表 Web 站点的根，即对应实际的站点主目录，主目录下的文件可以通过实际的相对路径访问。如果要访问主目录以外的文件，则需要使用虚拟目录。虚拟目录可以简化站点的管理，这些内容可以被删除而不会影响原有的站点结构。

例如，在本地计算机上的 Web 站点上，建立一个名称为 hao 的虚拟目录，对应实际物理目录是 d:\hao，里面包含文件 haoHome.htm，要浏览该网页，在地址栏中输入：http://

127.0.0.1/hao/haoHome.htm 即可。

#### 4. 文件夹和文件的命名

在开发实践中,为了管理的方便,在命名文件和文件夹时,需要遵循下面几条一般性的命名原则:

(1) 使用名称前缀。因为文件和文件夹的列表通常按照字母顺序,因此,可以给功能相近的文件夹或文件使用相同的名称前缀,从而保证列表时能够挨在一起。

(2) 使用名称后缀或序号。许多功能可以分成几个步骤,每一个步骤可能是一个网页文件,为了管理方便,在命名这些网页文件时,可以在名称后面部分添加序号或后缀,这样可以保证在列表中,这些文件是顺序相连的。

例如,注册一个用户,可能分成两个步骤,对应的网页文件名可以为:newuser(注册信息输入页面)、newusersave(数据库操作),也可以命名为 newuser1、newuser2 等。

(3) 大小写问题。有的 Web 服务器(例如 Tomcat)区分文件夹和文件名大小写,命名时要注意。

(4) 避免中文命名。因为有些 Web 服务器对中文命名支持不好,在命名文件夹和文件时,尽量避免中文名。

### 2.3.4 连接到 Web 站点

Web 站点是由一系列的文件夹构成的,每个文件夹中包含了一系列的文件和数据。每个 Web 站点都有一个主目录文件夹,该文件夹中包含站点的首页文件。要连接到一个 Web 站点,应该在浏览器地址栏中输入 Web 站点的 URL,一般形式为:

```
http://网址:端口号/路径/文件名? 参数表
```

其中:

(1) 网址:可以是域名,也可以为 IP 地址。端口号对应 Web 服务器上设定的 Web 站点 TCP 端口,默认值为 80。如果端口号为 80,则在 URL 中可以省略不写。

(2) 路径:是指相对于 Web 站点主目录的相对路径,如果不指定路径,则代表站点主目录。

(3) 文件名:访问一个 Web 站点,即从 Web 站点中指定的路径中下载文件,并传输到客户端浏览器进行显示的过程。因此,在 URL 中需要指定要下载文件的路径和文件名。如果未指定文件名,则代表要下载网站首页文件,首页文件在 Web 站点属性中设置,并存储在 Web 站点根目录中。

(4) 参数表:在访问一个网页文件时,特别是带有脚本的网页,有时候需要将一些参数传给网页中的脚本程序,这些要传递的参数在文件名后面的"?"后面列出。

如果是在 IIS 服务器计算机上,也可以输入 http://127.0.0.1 或 http://localhost 来访问本机上的 Web 站点。其中,localhost 为本机(127.0.0.1)的域名。可以用记事本打开 hosts 文件(文本文件,无扩展名,存储在\WINNT\system32\drivers\etc 文件夹中)看到 127.0.0.1 的域名为 localhost。

# 2.4　Web 站点的配置

当 Web 站点建立后，还需要对 Web 站点进行管理，管理 Web 站点是通过 Web 站点属性对话框来完成的。在“Internet 信息服务管理器”控制台目录树中，右击站点，执行“属性”命令，打开站点属性对话框，通过站点属性对话框，可完成一个站点的配置和管理。

## 2.4.1　设置 Web 站点端口号

在 Web 站点属性对话框中，“网站”选项卡列出了网站的一般属性，默认值为创建站点时的用户输入，如图 2-14 所示。

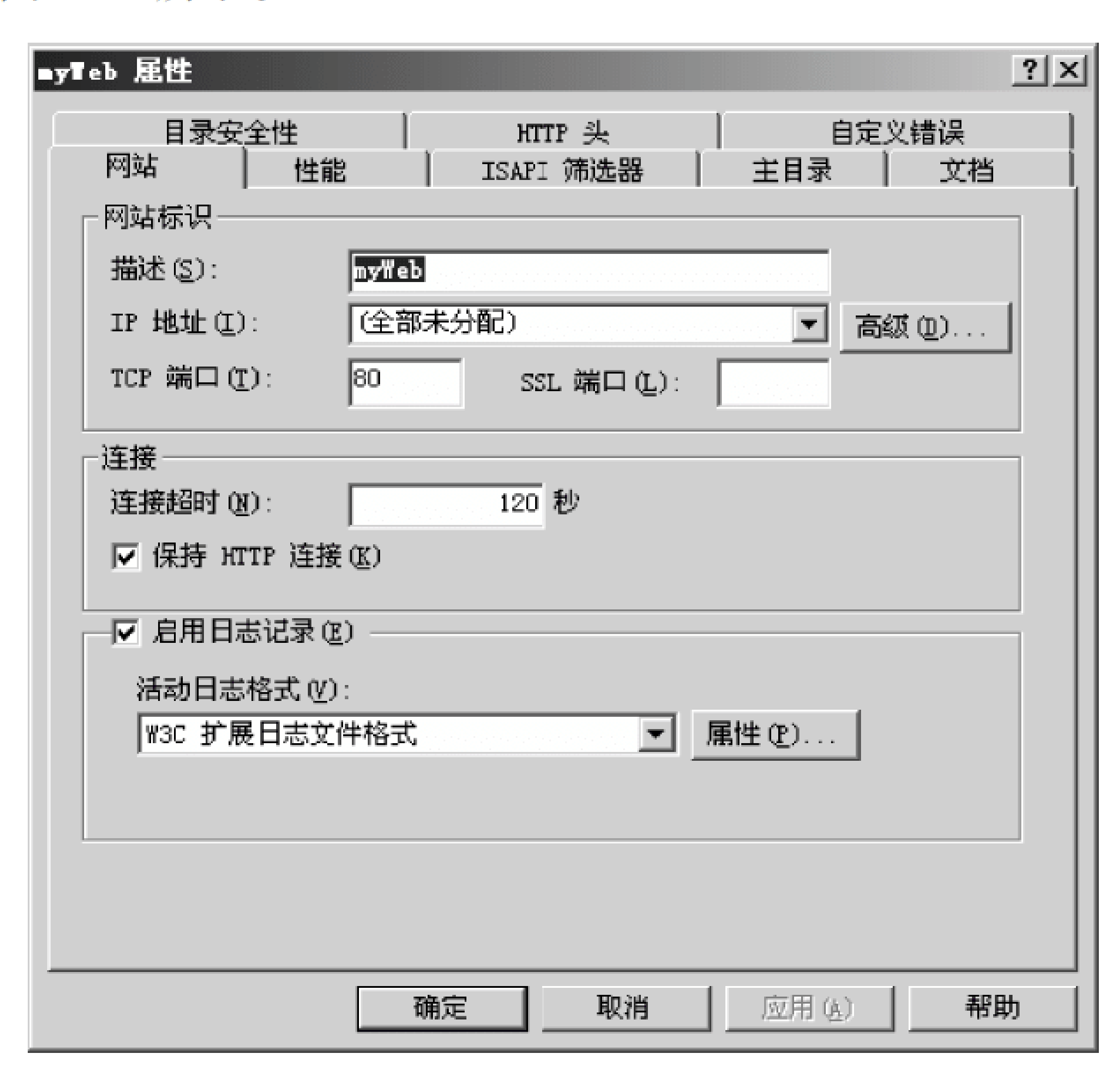



图 2-14　Web 站点属性对话框

在 Web 站点选项卡中，包括三个区域的设置。

### 1. 网站标识

- 说明：输入对该站点的说明性文字，该文字将作为站点名字出现在 Internet 信息服务管理器控制台目录树中。
- IP 地址：设置此站点要使用的 IP 地址，如果计算机中设置了多个 IP 地址，可以选择其中一个。如果该服务器上同时运行多个 Web 站点，单击“高级”按钮，可进行进一步的设置。
- TCP 端口：HTTP 服务的默认端口为 80，如果设置其他端口，例如将端口号设置为 8001，则客户要访问该网站时，在浏览器地址栏中需要给出端口号，即 http://网址:8001/。

#### 2. 连接

- 限制到：当 IIS 服务器的内存和网络带宽较小时，可以限制该 Web 站点可以连接的客户数量。选择"限制到"单选钮，可以指定该站点最多的连接数量。
- 连接超时：如果客户端建立了连接，在连接超过规定的时间内没有访问操作，系统将该连接强制断开。
- 启用保持 HTTP 激活：如果一个网页中插入了其他文件(如图片、动画等)，让网页和其中的文件通过一个连接传送，从而降低 Web 站点的负担。

如果清除"启用保持 HTTP 激活"复选框，当网页中包含多个文件连接时，客户端每下载一个文件就要与 Web 服务器建立一个连接，将降低 Web 服务器的执行性能。

#### 3. 启用日志记录

选择该选项将启用 Web 站点的日志记录功能，该功能可记录用户活动的细节并以选择的格式创建日志。启用日志记录后，需要在"活动日志格式"列表中选择格式。

### 2.4.2 设置 Web 站点主目录

主目录是一个网站的根，网站的所有文件都保存于主目录及其所包含的子文件夹中，或者通过虚拟目录使用主目录外的物理文件夹。根据客户访问 Web 站点的验证过程，当用户通过身份验证后，接下来，Web 站点会根据站点的权限设置来决定可以提供给用户的服务，例如从网站浏览网页(下载文件)、上传文件等。

在网站属性对话框中，单击"主目录"选项卡，如图 2-15 所示。

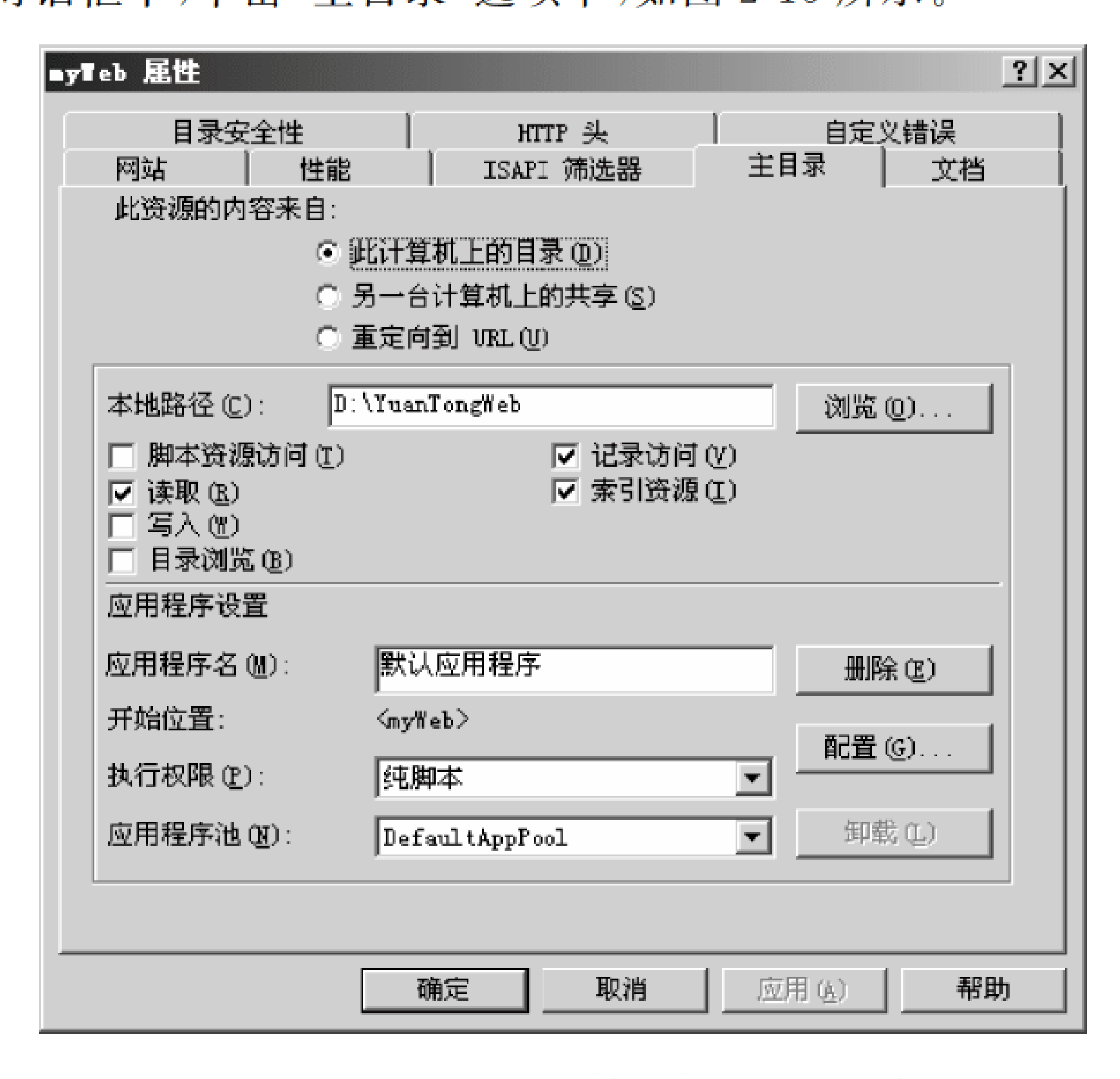



图 2-15 网站属性对话框"主目录"选项卡

### 1. 访问权限设置

- 读取：默认状态下 Web 站点拥有读取权限，即站点提供客户读取服务器上文件的权限，即客户可以从站点中下载文件。
- 写入：允许用户上传文件，或提交表单改变网页内容。
- 目录浏览：允许用户浏览站点目录，当客户通过浏览器连接到本站点时，如果未指定文件名和目录，站点也没有启用默认文档，或默认文档不存在，将看到此站点的目录列表（不显示虚拟目录），如图 2-16 所示。




图 2-16　网站目录浏览界面

### 2. 应用程序设置

应用程序设置可以指定何种应用程序可以在 Web 站点执行，在执行许可列表中，包括“无”、“纯脚本”和“脚本和可执行程序”。

如果选择“无”，则不允许在 Web 站点中运行程序（包括服务器端 ASP 脚本），当浏览一个 ASP 页时，会显示“网页无法显示”，在页面中提示：“您试图从目录中执行 CGI、ISAPI 或其他可执行程序，但该目录不允许执行程序”。

选择“纯脚本”，则只能执行 ASP 程序等。选择“脚本和可执行程序”，则所有的应用程序（包括 exe 文件和 dll 库）都可以在 Web 站点上执行。

## 2.4.3　Web 站点目录安全性配置

Web 站点的安全性设置主要是通过“目录安全性”选项卡完成的，在介绍具体的安全性设置以前，先来介绍 Web 站点的访问机制。

当客户端通过浏览器向 Web 站点发出访问某个页面的请求时，Web 服务器收到客户

的请求后，将启动一个验证过程，来决定是否将网页传给客户端，验证过程如图 2-17 所示。

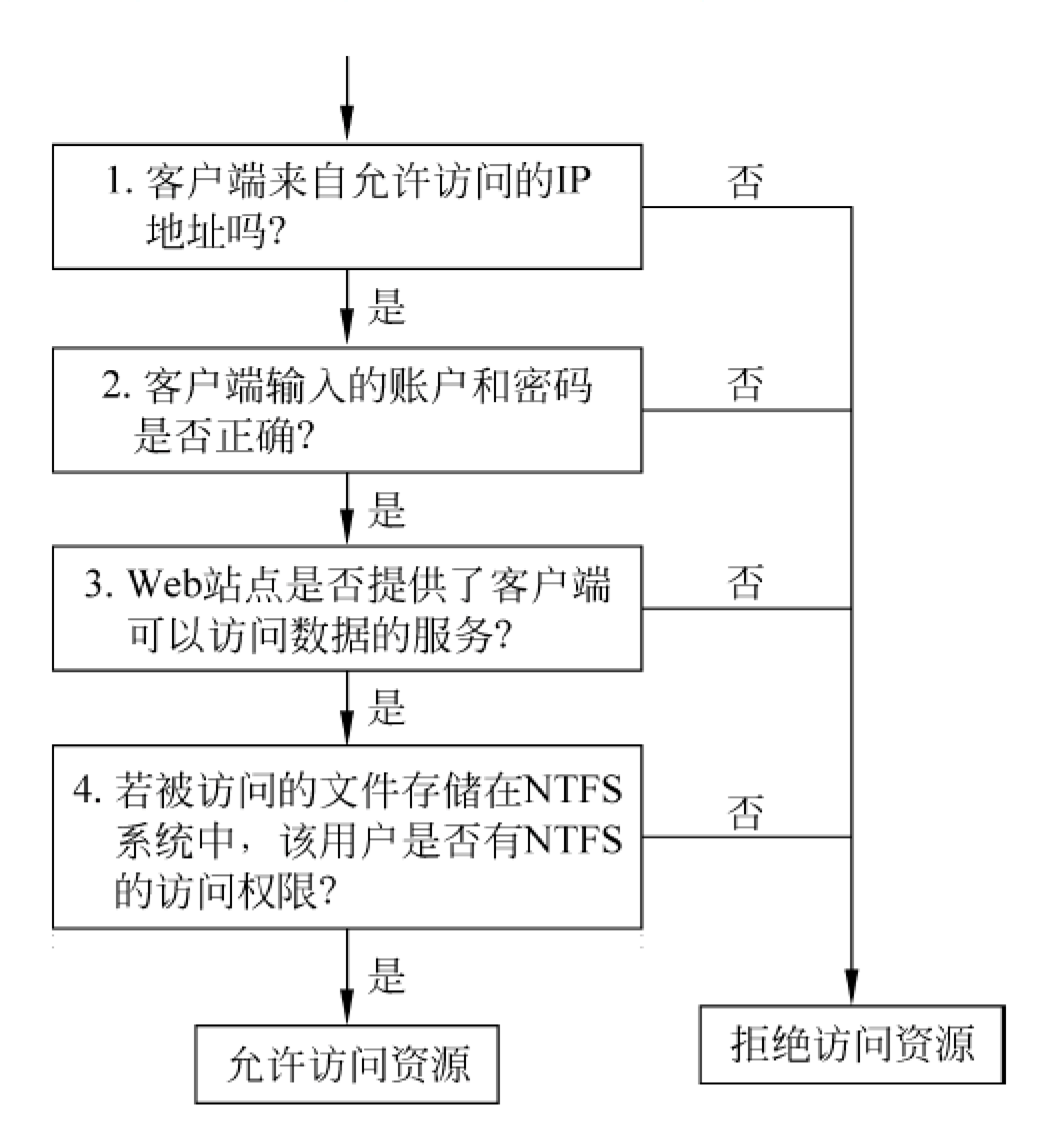



图 2-17　客户访问请求验证过程

通过验证过程的验证后，如果网页是 HTML 类型的，则 Web 服务器将把该网页直接传送到客户端浏览器。如果网页为 ASP、JSP 等含有服务器脚本的文件，Web 服务器将先在服务器端执行该网页文件，然后将执行结果网页传给客户端浏览器。

要进行 Web 站点的安全性设置，在站点属性对话框中，单击"目录安全性"选项卡，如图 2-18 所示。

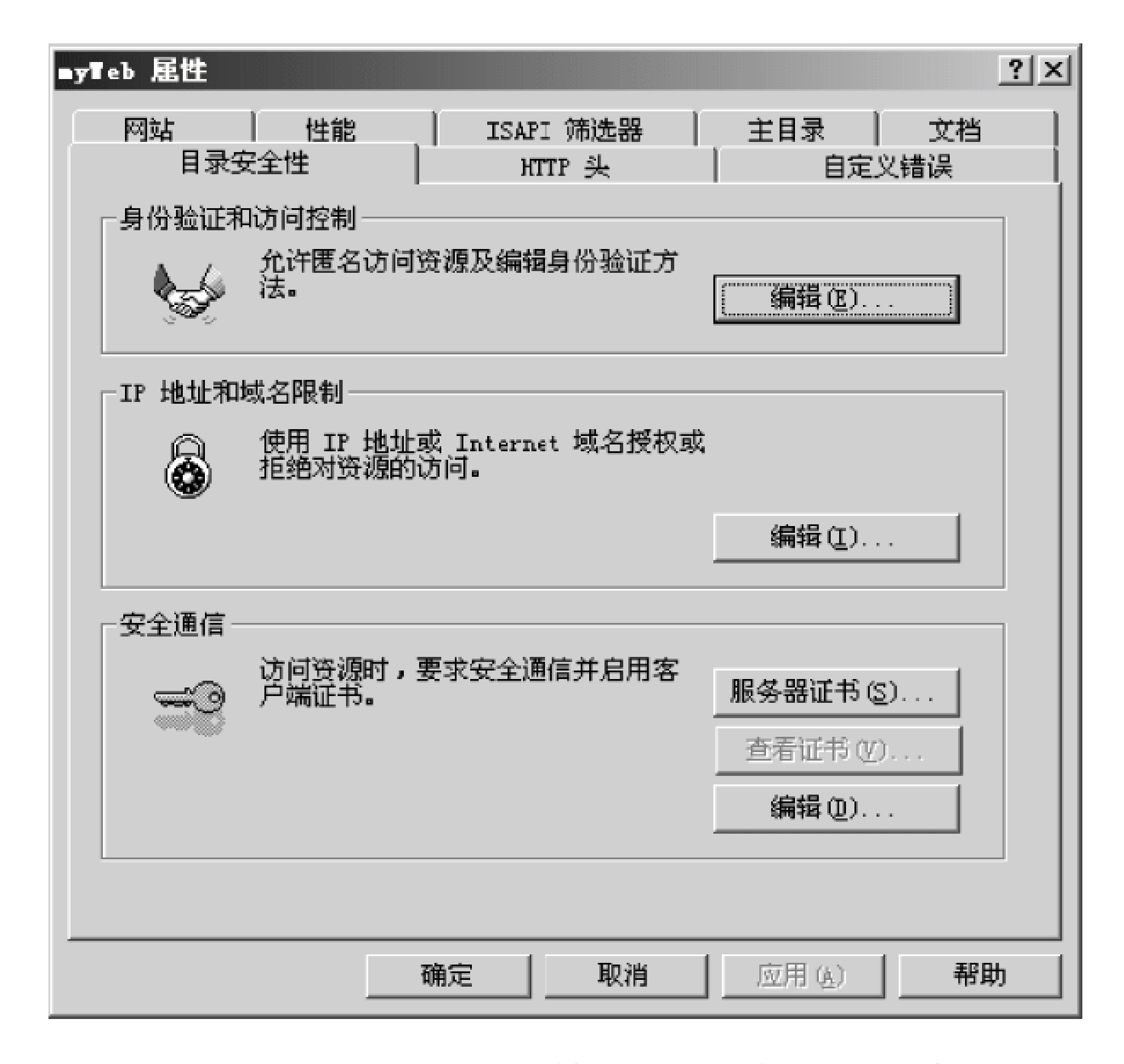



图 2-18　网站属性对话框"目录安全性"选项卡

### 1. IP 地址及域名限制

在 IP 地址及域名限制区域中，单击“编辑”按钮，如图 2-19 所示。

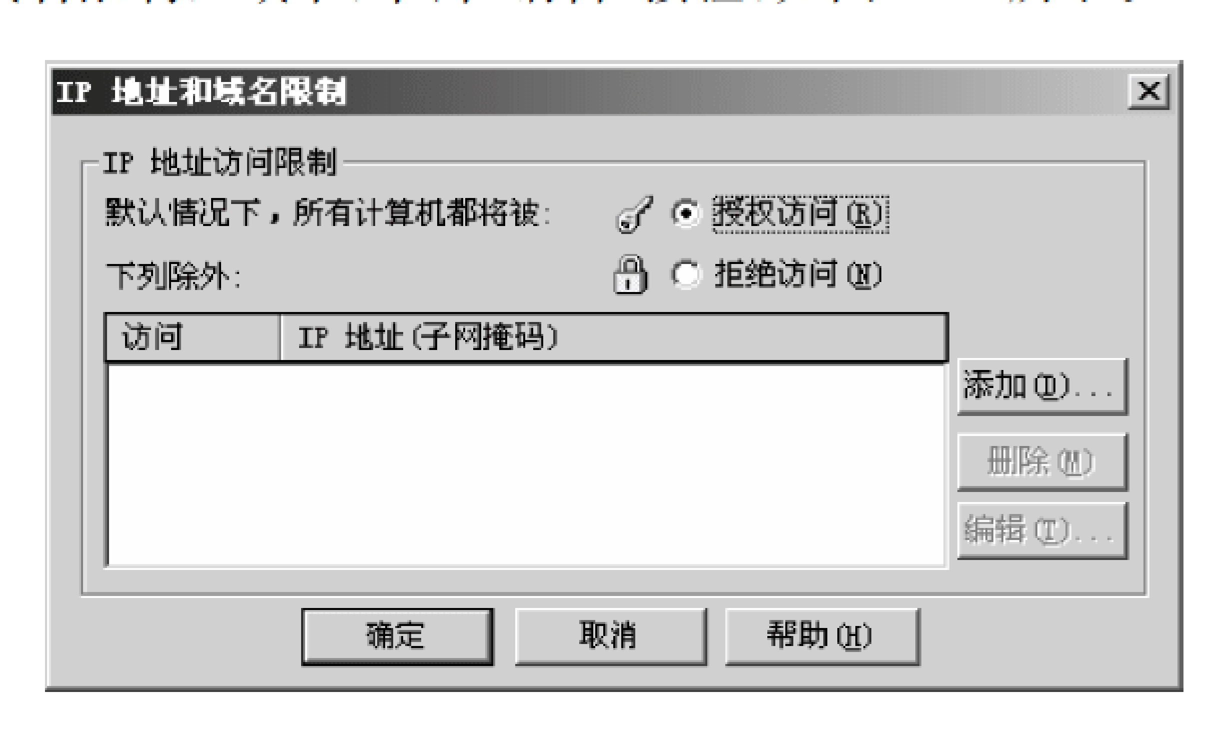



图 2-19　“IP 地址和域名限制”对话框

在“IP 地址和域名限制”对话框中，选择“授权访问”，然后单击“添加”按钮，可以指定不能访问该站点的 IP 地址。类似地，选择“拒绝访问”单选钮，通过添加，可以指定在拒绝访问中，能够访问该站点的 IP 地址清单。

当用户来自拒绝访问的 IP 地址时，客户浏览器端会收到“您没有权限查看网页”的提示信息。一般情况下，如果网站是公开的，一般选择“授权访问”，然后单击“添加”按钮，把不被欢迎的 IP 地址列出。相反，如果网站是一个特殊的站点，只允许部分人访问，则选择“拒绝访问”，然后把可以访问的 IP 列出。

### 2. 匿名访问和验证控制

当 Web 站点验证了客户端的 IP 地址后，接下来查看该站点是否允许匿名访问。如果站点不允许匿名访问，或者客户端要访问的文件有特殊的 NTFS 限制，此时客户端需要输入用户账户和密码。

当 Web 站点允许匿名访问时，客户端不需要输入账户和密码就可以访问网站的数据，此时 Web 站点会尝试用 Internet Guest Account 账号“IUSER_计算机名称”这个内部账户让计算机登录。要设置匿名访问，在“匿名访问和验证控制”区域中，单击“编辑”按钮，打开“身份验证方法”对话框，如图 2-20 所示。

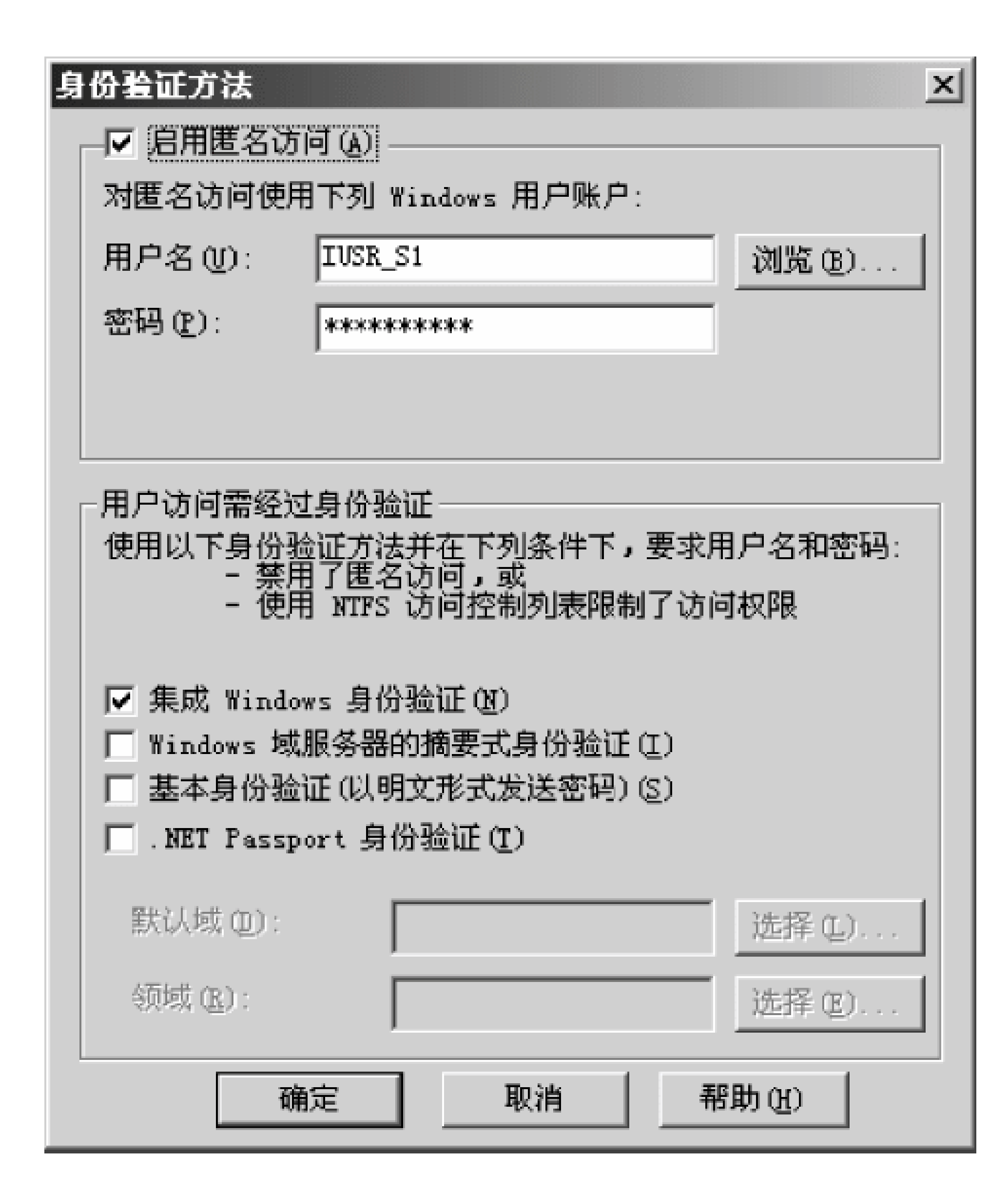



图 2-20　“身份验证方法”对话框

选择“匿名访问”。单击“编辑”按钮，打开“匿名用户账号”对话框。在该对话框中，可以指定用于匿名访问的匿名用户账号。匿名访问使得每个人都可以使用上述账号访问 Web 网

站。如果匿名账户没有足够的 NTFS 权限,系统会根据在“验证访问”区域中选择的验证方式,要求用户输入账号和密码,如果未选择任何验证方法,则系统不提示用户输入账户和密码,而是直接拒绝用户对该页的访问。

一般情况下,如果 Web 站点连接到 Internet,一般选择“启用匿名访问”复选框,即允许匿名访问。

#### 3. 使用权限向导

如果 Web 站点设置了匿名访问,当客户访问该站点时,仍然出现“输入网络密码”对话框,这是由于匿名账户不拥有要访问的 NTFS 权限造成的。

为了避免遗漏对 Web 主站点的 NTFS 权限设置,导致客户端不能正常地访问所需要的 Web 页,在 Internet 信息服务控制台中,提供了“权限向导”命令。右击站点,在快捷菜单中指向“所有任务”,单击“权限向导”,启动“IIS 权限向导”。按照向导提示操作,一般取默认值,最后单击“完成”按钮。再来访问该站点,看能否成功。

### 2.4.4 设置 Web 站点默认文档

下面介绍如何设置站点的默认文档,即相当于站点的首页。默认文档可以是 HTML 文件,也可以是 ASP、JSP 等包含服务端脚本的文件。当用户通过浏览器连接到 Web 站点时,如果没有指定要浏览的文档,Web 站点则将默认文档传送给用户浏览器。

在 Web 站点属性对话框中,选择“文档”选项卡,如图 2-21 所示。

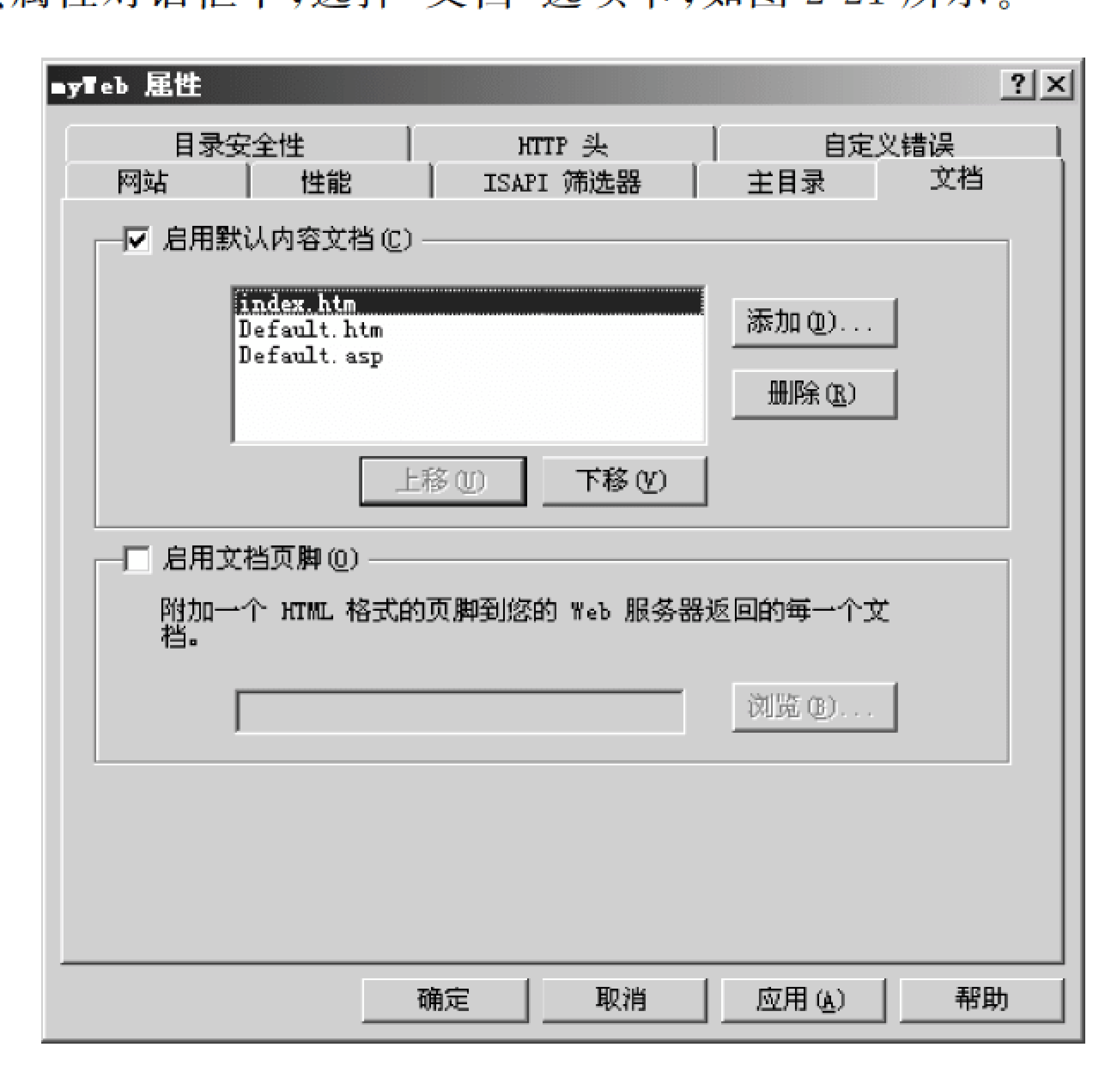



图 2-21 网站属性对话框“文档”选项卡

选择“启用默认内容文档”复选框,用户也可以单击“添加”按钮,增加一个新的默认文档,如 index.htm、startpage.htm 等。

如果有多个默认文档，系统将把排在前面的文档优先传送给客户浏览器。

如果选择“启用文档页脚”复选框，则服务器在传送要求的网页之前，会在文档的底部插入页脚文字，然后再传送。

文档页脚对应一个 HTML 文件，这个 HTML 文件不应该是一个包含<html></html>、<body></body>等标记的完整的 HTML 文件，只能包含文字的大小和颜色设置即可，如：

```
<h3 align = right>E - learning 站点</h3>
```

文件可以用 Windows 操作系统中的“记事本”程序编辑，并保存为 .htm 类型的文件。

### 2.4.5 设置 Web 站点 HTTP 头

HTTP 头（HTTP Header）是对现有 HTTP 标准的扩充，有许多复杂的应用，在“HTTP 头”选项卡中，可以对站点做 4 种设置，如图 2-22 所示。




图 2-22　网站属性对话框“HTTP 头”选项卡

选择“启用内容过期”复选框，可以设置此站点内容到期的时间。当用户浏览一个站点的某个网页时，服务器首先将浏览器要访问的 Web 页的 URL 返回到客户端，客户端在本地硬盘的网页缓存中查找是否存在该页面，如不存在，将要求服务器传送该页面。否则，浏览器将对要下载 Web 页的当前日期和到期日期进行比较，来决定是显示客户端硬盘中网页缓存的页面，还是向 Web 站点更新新的网页。

选择“立即过期”，则网页内容一下载到浏览器端该页面就过期了。因此，浏览器每次连接到该网站时，无论客户端的本地网页缓存是否存在对应的页面，页面都会被重新下载。它适合于一些显示即时行情的网站，如：股市行情。

选择“此时间段后过期”,用于设置网页的有效期,当浏览器连接到该站点浏览网页时,网页被保存在客户端的缓存文件夹中,时间到后,该网页将自动地从客户端缓存中删除,适合于一些固定时间更新的新闻站点和页面。

## 2.5 使用 Apache 和 Tomcat

在 Web 服务器产品中,主要有 Windows 平台中的 IIS 和 Apache/Tomcat,其中后者也可以安装在 Linux、UNIX 等操作系统平台中。Web 服务器决定了 Web 程序的类型,IIS 支持 ASP 和.net 开发,如果需要开发基于 Java 技术的网站,则需要安装 Apache/Tomcat 服务器。

在 Windows 平台中,相比 IIS,Apache/Tomcat 服务器的安装、配置和管理相对要复杂一些,但有关网站的基本概念是一样的。此外,在 Windows 平台中,如果已经安装了 Internet 信息服务 IIS,要使用 Apache 和 Tomcat,应将 Windows 中的 IIS 服务停止或禁用。

### 2.5.1 Apache 与 Tomcat

Apache 是使用最广的 Web 服务器之一,有多个操作系统平台版本,它可以运行在几乎所有广泛使用的计算机系统平台上,以高效、稳定、安全、免费而著称。作为 Web 服务器,Apache 服务器具有以下特性:

- 支持最新的 HTTP/1.1 通信协议。
- 拥有简单而强有力的基于文件的配置过程。
- 支持通用网关接口 CGI。
- 支持基于 IP 和基于域名的虚拟主机。
- 支持多种方式的 HTTP 认证。
- 集成 Perl 处理模块。
- 集成代理服务器模块。
- 支持实时监视服务器状态和定制服务器日志。
- 支持服务器端包含指令(SSI)。
- 支持安全 Socket 层(SSL)。
- 提供用户会话过程的跟踪。
- 支持 FastCGI。
- 通过第三方模块可以支持 Java Servlet。

Tomcat 是针对于 Apache 服务器开发的 JSP 应用服务器,是 Java Servlet 和 Java Server Pages(JSP)技术的标准实现,是基于 Apache 许可证下开发的自由软件。可以从网站 http://jakarta.apache.org/tomcat/index.html 下载不同的 Apache Tomcat 版本。

可以这样认为,当在一台机器上配置好 Apache 服务器,可利用它响应对 HTML 页面的访问请求。实际上 Tomcat 部分是 Apache 服务器的扩展,但它是独立运行的,所以当运行 Tomcat 时,它实际上作为一个与 Apache 独立的进程单独运行。当配置正确时,Apache 为.html 页面服务,而 Tomcat 实际上运行.jsp 页面和 Servlet。

要在 Apache 下运行 JSP,最好的方案就是选择 Tomcat,它具有免费、集成度好的优点,

缺点是界面不够直观，如果是在 Windows 平台上，还需要设置环境变量，相对麻烦。

### 2.5.2 Apache 的安装和配置

Apache 服务器为开源软件，可以从 Apache 官方网站(http://www.apache.org/)下载。在 Apache 官方网站首页中，有一个"Apache Projects"列表，显示 Apache 项目超链接列表，单击"HTTP Server"超链接，将打开 Http Server 项目页面(http://httpd.apache.org/)。

#### 1. 下载 Apache 服务器

在 Http Server 项目页面(http://httpd.apache.org/)，选择要下载的 Apache 版本。需要说明的是，版本不一定是最新的，但一定要选择一个稳定的版本，目前使用较广的版本是 Apache HTTP Server 2.2.8。然后，根据操作系统不同选择不同的 Apache 服务器。

对于 Windows 平台，有两个可选的版本，分别是 Win32 Binary without crypto(no mod_ssl)和 Win32 Binary including OpenSSL 0.9.8g。OpenSSL 为开放安全套接层协议(Secure Socket Layer，SSL)，可以在 Internet 上提供秘密性传输，包含密码算法库、SSL 协议库以及应用程序，目前的版本为 0.9.8g。选择 Win32 Binary including OpenSSL 0.9.8g 下载，将服务器文件 apache_2.2.8-win32-x86-openssl-0.9.8g.msi 下载到本地计算机(4.8M)。

目前，Apache for win32 使用 msi 的形式发布，.msi 文件类型是一种可以安装的程序包文件，双击带.msi 扩展名的文件时，操作系统将.msi 文件与 Windows 安装程序关联并运行客户端安装程序服务 Msiexec.exe，从而使 Windows 环境下安装 Apache 变得非常简单。

#### 2. Apache 的安装

当 Apache 服务器下载后，可以按照下列步骤完成 Apache Web 服务器的安装和配置。

(1) 双击 Apache 的安装文件 apache_2.2.8-win32-x86-openssl-0.9.8g.msi，执行安装向导，如图 2-23 所示。




图 2-23 Apache 服务器的安装

(2) 单击 Next 按钮,按照向导提示,分别输入 Network Domain(网络域名,如：xxx.com)、Server Name(服务器域名,如：www.xxx.com)和网站管理员的 E-mail,按照网站的实际情况填写,如果是个人用户,可能没有上述数据,可按格式填一下临时的名字。如图 2-24 所示。

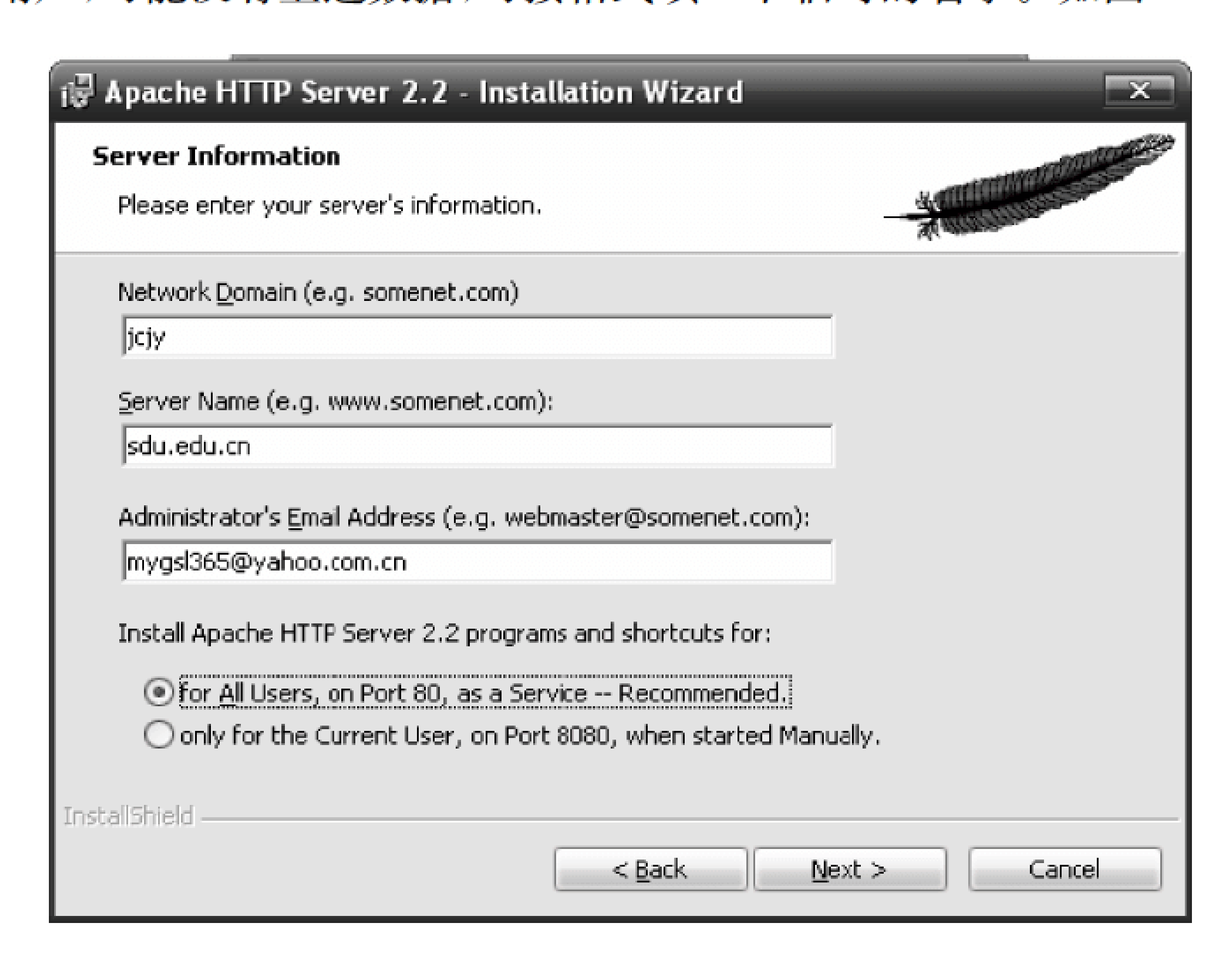



图 2-24　Apache 安装向导输入信息屏幕

在 Apache 的安装过程中,需要输入网站域名,如果仅仅是本地调试,使用 localhost 即可。然后,单击 Next 按钮,选择安装类型(Typical 或 Custom)。

单击“Typical”安装,按照向导提示操作,选择安装路径,直至安装完成。

Apache 服务安装成功后,在 Windows“开始”菜单中增加“Apache HTTP Server 2.2”程序组。同时,在控制面板、管理工具文件夹下,双击“服务”图标,显示 Apache 已经启动,以后 Apache 将作为一项服务,随着机器的启动而自动运行。

不需要重新开机,Apache 会自动启动,在 Windows 任务栏的右侧显示“Running all Apache Services”图标。此时在 IE 地址栏里输入：http://localhost 或 http://127.0.0.1 看到默认的 Apache 首页,显示“It works”。

需要说明的是,如果计算机上已经安装了 IIS,输入 http://localhost 后将首先显示 IIS 的默认站点。此时,可以按照第 2.3.2 节的介绍停止 IIS 中的 Web 服务器,或者给 IIS 中的 Web 服务指定一个不同的端口号(不同于默认的 80 端口)。然后再输入 http://localhost 即可显示 Apache 服务器设置的页面。此时还可以通过 http://localhost：端口号/继续打开 IIS 中的默认 Web 站点。

### 3. Apache 的配置

Apache 的主配置文件为纯文本格式的 httpd.conf,默认情况下,它的存储位置为 C:\Program Files\Apache Group\Apache\Conf\。随着 Apache 版本的发展,趋向于使用单一的配置文件 httpd.conf 来存放所有的配置指令,如客户访问信息、记录认证信息和虚拟服务器信息等等。

Apache 配置选项采用的是指令模式,配置指令设定各种参数的值,例如：DocumentRoot

设置服务器 Web 页面的根目录。也可以灵活地设置多个基于 IP 或基于域名的虚拟 Web 服务器，这些 Web 虚拟服务器可以各自定义独立的 DocumentRoot 配置指令。而 LoadModule 指令则用来指定加载不同的模块来实现对 Apache 服务器功能的扩充。这些新功能大多是提供服务器端对脚本技术的支持，比如 Perl、PHP 等。Apache 结合使用 ApacheJServ 可以实现对 Java Servlet 及 JSP 的支持。

用记事本打开它，可以看到这些配置文件都以文本方式存在，其中“#”为 Apache 的注释符号，我们可以在记事本菜单中的编辑选项中单击“查找”逐一输入下面要配置的关键字，并进行相应配置。此外，打开 Windows 的“开始”菜单，执行“程序”、“Apache HTTP Server 2.2”、“Configer Apache Server”、“Edit the Apache httpd.conf Configuration File”命令，将打开记事本，显示“httpd.conf”文本文件，进行 Apache 的配置。

1）配置 DocumentRoot

这个语句指定网站路径，即主页放置的目录。默认路径一般是 Apache 安装目录下的一个子目录，例如：

```
DocumentRoot "C:/Program Files/Apache Software Foundation/Apache2.2/htdocs"
```

根据需要，设置站点的主目录，例如我们可以在此处将其设定为“D:/GSL3.0”，打开主页时，默认打开的文档就直接去该目录下查找了。

2）配置 DirectoryIndex

这是站点默认显示的主页，一般情况下，我们在此处还可以加入“Index.htm Index.php Index.jsp”等。注意，每种类型之间都要留一空格。

上面两步设置完成后，启动浏览器，输入 IP 即可访问自己的 Web 站点。还可以在该文件的 ServerName 处定义域名，在 ServerAdmin 处输入 E-mail 地址。以上两条就是在安装时选择配置的，以后可以在此处修改它们的属性。

此外，如果要拒绝一部分人访问该 WWW 站点，可以到 Apache 的安装目录下找到 Access 文件，输入要禁止的 IP 地址即可。

### 2.5.3　Tomcat 服务与 Servlet/JSP 规范

Tomcat 是当前使用最为广泛的 Servlet/JSP 服务器，它是 JavaSoft 和 Apache 开发团队合作计划（Apache Jakarta Project）的产品，被 Sun 公司作为官方推荐的 Servlet 和 JSP 容器，具有运行稳定、性能可靠、免费的特点，是学习 Web 开发的最佳选择。

随着 Sun 公司推出的 Servlet/JSP 规范的不断完善和升级，Tomcat 的版本也随之不断更新，使得 Tomcat 的版本很多且复杂。Tomcat 版本和 Servlet/JSP 规范的对应关系见表 2-1。

**表 2-1　Tomcat 版本和 Servlet/JSP 规范的对应关系**

| Apache Tomcat 版本 | Servlet/JSP 规范 | 主要产品 |
|---|---|---|
| 6.0.x | 2.5/2.1 | Tomcat 6.0.14 |
| 5.5.x | 2.4/2.0 | Tomcat 5.5.12 |
| 4.1.x | 2.3/1.2 | |
| 3.3.x | 2.2/1.1 | |

可以从 Apache 网站 http://tomcat.apache.org/下载所需要的 Apache Tomcat 版本，本章选择最新的 apache-tomcat-6.0.14.exe(集成实现了 Servlet2.5 和 JSP2.1 标准)，讲解其具体的安装和配置。

### 2.5.4 安装 Java 运行环境

Tomcat 需要 Java VM(JRE)(Java Runtime Environment)，即 Java 虚拟机的支持，因此，在安装 Tomcat 以前需要安装 JRE。JRE 可以单独安装，也可以随 Java 开发包 JDK 一起安装。安装 JRE 后，在安装 Tomcat 时会自动监测到。

Java 技术中的 Java 运行环境包括两个主要的部分：Java 开发工具包和 Java 运行环境 JRE。它们是基于 Java 技术开发和运行的基础环境。在 Windows 平台上，Java 环境安装完成后需要手工进行相应的环境变量配置，方能正确地工作。

#### 1. 什么是 JDK 和 JRE

在安装 Java 环境以前，需要介绍几个概念。在 Java 技术中，大家经常看到 JDK、J2SDK 和 JRE 等概念，有时候会产生迷惑，三者是一种什么关系呢？

JDK 是 Sun 早期的 Java 软件开发工具包(Java Develop Kit，JDK)，包含了所有编写、运行 Java 程序所需要的工具：Java 基本组件、库、Java 编译器、Java 解释器、小应用程序浏览器以及一些用于开发 Java 应用程序的程序等。从 JDK1.2 起，Sun 在命名时开始使用 Java 2，这就是 J2SDK 了，又分为企业版(Enterprise Edition)J2EE、标准版(Standard Edition)J2SE 以及面向嵌入式和移动计算等领域的 J2ME(Micro Edition)三个不同的版本，详细说明可参见第 1.4.2 节“Java 技术”的介绍。

JRE(Java Runtime Environment)，顾名思义是 Java 程序运行所需要的环境。所谓跨平台就是要各种平台都有一个中间代理，这就是 JRE。一般采用 Java 技术开发出的软件都需要安装 JRE，所以 Sun 就单独提供了 JRE 安装文件，以供 Java 应用程序发布时所用。

以上 Java 软件都可以从 Sun 的 Java 网站(http://java.sun.com)上获取，网站上分别提供了 J2EE SDK、J2SE SDK 以及 Java VM(JRE)各种版本的下载。

#### 2. 安装 JDK 和 JRE

Sun 公司网站(http://java.sun.com/)提供了 J2SDK 和 JRE 的集成安装和单独安装文件，用户可以免费下载。目前较新，同时比较稳定的版本是 JDK6。根据开发和应用的不同，可以选择企业版或标准版，我们以 J2SE6 为例，介绍 JDK 和 JRE 的安装过程。

首先，登录 Sun 官方网站 http://java.sun.com/，在常用现在区域(Popular Downloads)单击“Java SE”超链接，显示 Java JDK 和 JRE 下载界面，选择“JDK 6 Update 3”，下载文件为 jdk-6u3-windows-i586-p.exe，包含了 JDK6 和 JRE。

接下来进行 JDK6 和 JRE 的安装过程，双击 jdk-6u3-windows-i586-p.exe 文件，运行 JDK6 安装向导，显示许可协议，然后进行自定义安装界面，如图 2-25 所示。

按照向导提示将 Java 开发环境安装到计算机中，默认的文件夹为 C:\Program Files\java\jdk1.6.0_03。为了下一步环境变量设置的方便，通常需要修改默认安装目录，例如，

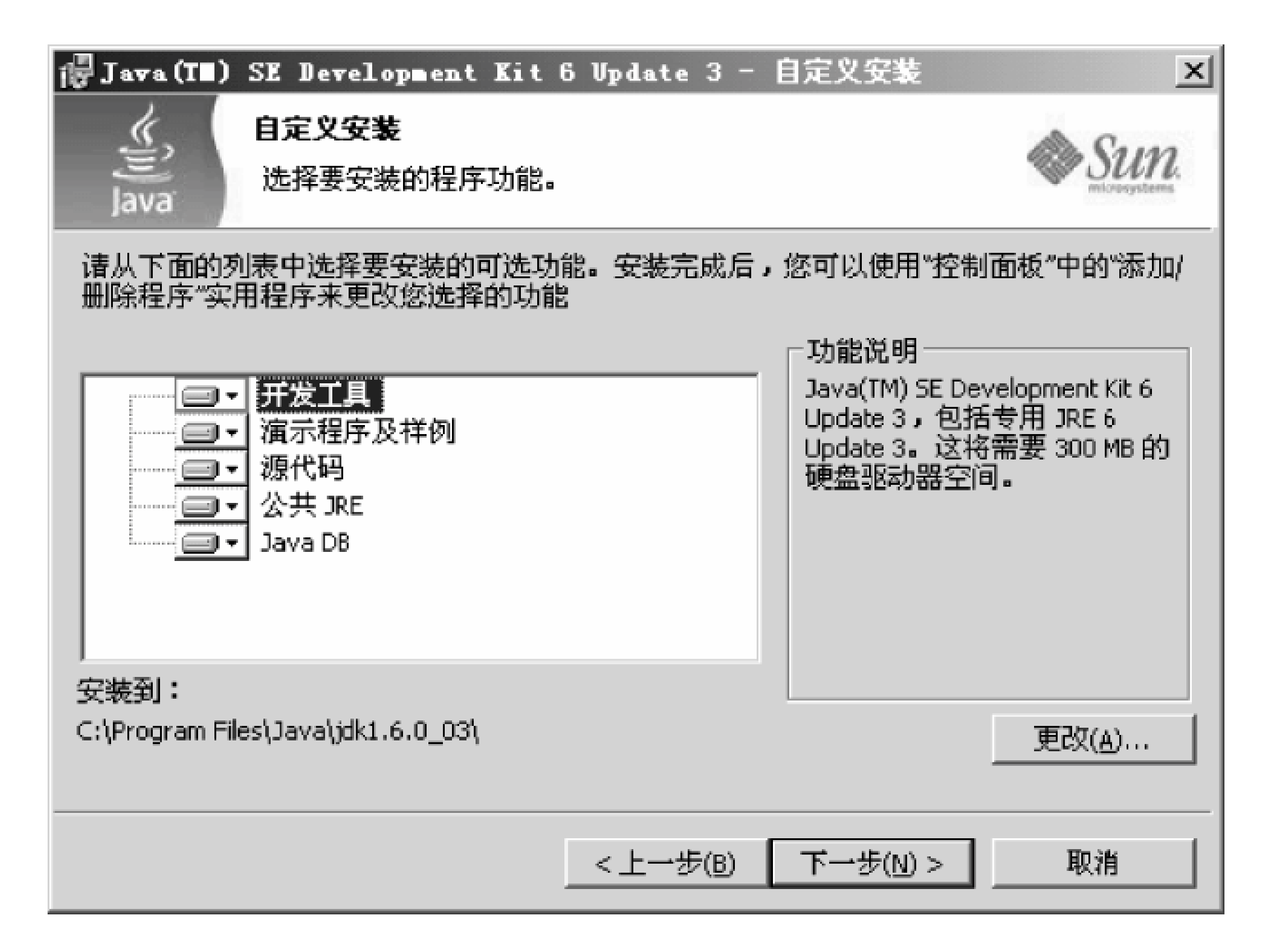



图 2-25　Java 2 SDK 标准版安装向导界面

直接安装在 C:\Java 目录下，即：C:\Java\jdk1.6.0_03\，这样可以便于环境变量的设置。

由于 jdk1.6.0_03 已经包含了 JRE，如果机器尚未安装 JRE，则在安装 jdk1.6 时，JRE 将一并安装。安装过程也需要指定安装路径。和安装 JDK 同样的原因，可以设置 JRE 的安装目录为 C:\Java\jre1.6.0_03\。

当 JDK 和 JRE 安装完成后，安装程序在 C 盘中建立相应的文件夹结构，存储相应的 Java 运行环境，文件夹结构如图 2-26 所示。

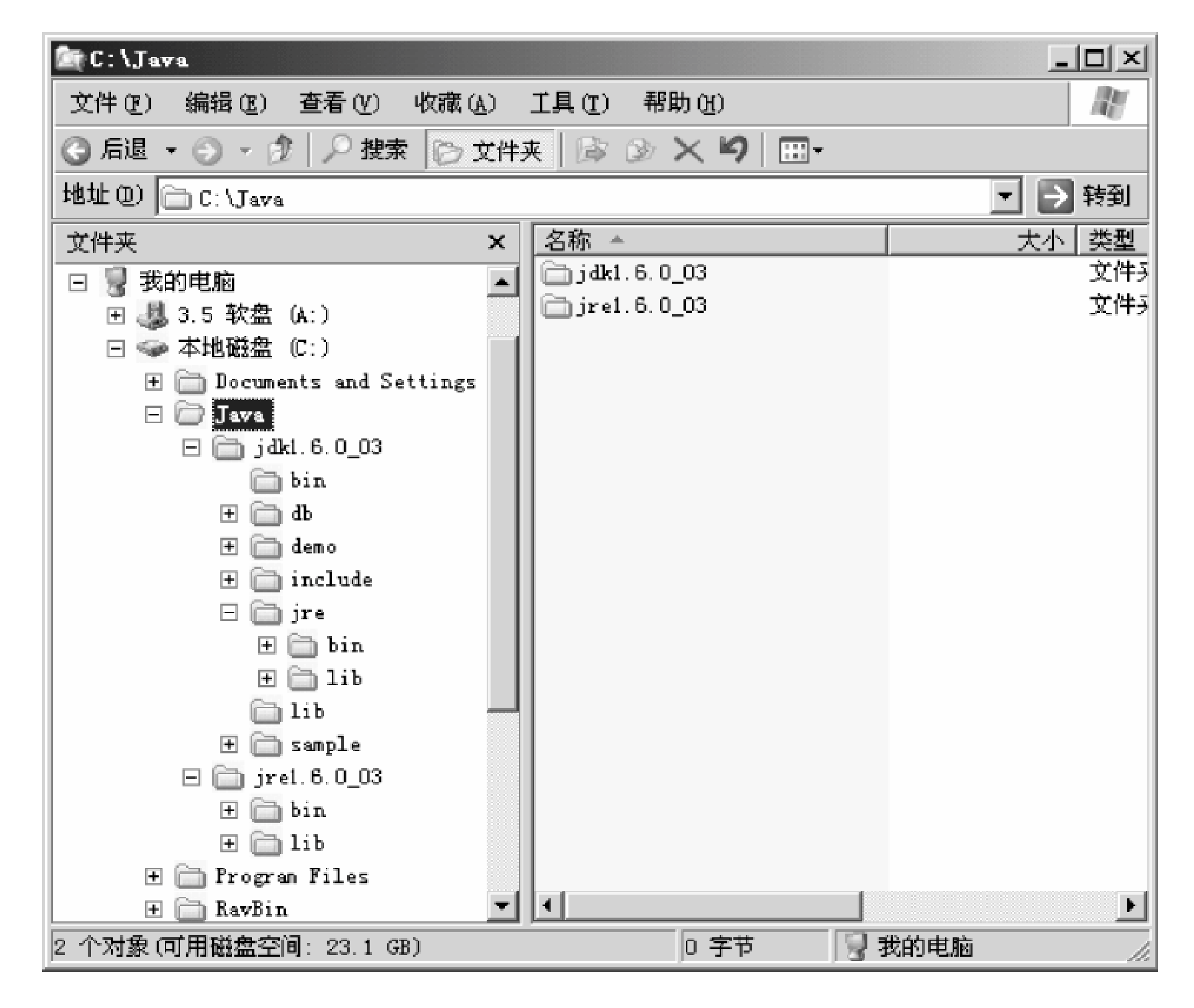



图 2-26　安装 JDK 和 JRE 文件夹结构

按照向导提示安装完成后，在“控制面板”中显示一个咖啡杯图标。双击该图标，将打开“Java 控制面板”。

用户可以通过“控制面板”中的“添加/删除程序”删除已经安装的 JDK/JRE。

### 3. Java 环境变量设置

JDK 安装完成后，需要进行相应的环境变量设置，以保证 Java 程序中对 JDK 中类库的引用。需要进行的环境变量设置包括设置 JAVA_HOME 和 CLASSPATH 环境变量、更新 PATH 路径设置三个部分。

为了检查 JDK 安装程序是否已经正确地设置了环境变量，可以使用 set <环境变量> 来检查，具体办法是：

在 DOS 提示符下，通过 set <环境变量>命令显示环境变量的配置情况。JDK6 安装完成后，环境变量设置检查结果显示如图 2-27 所示。

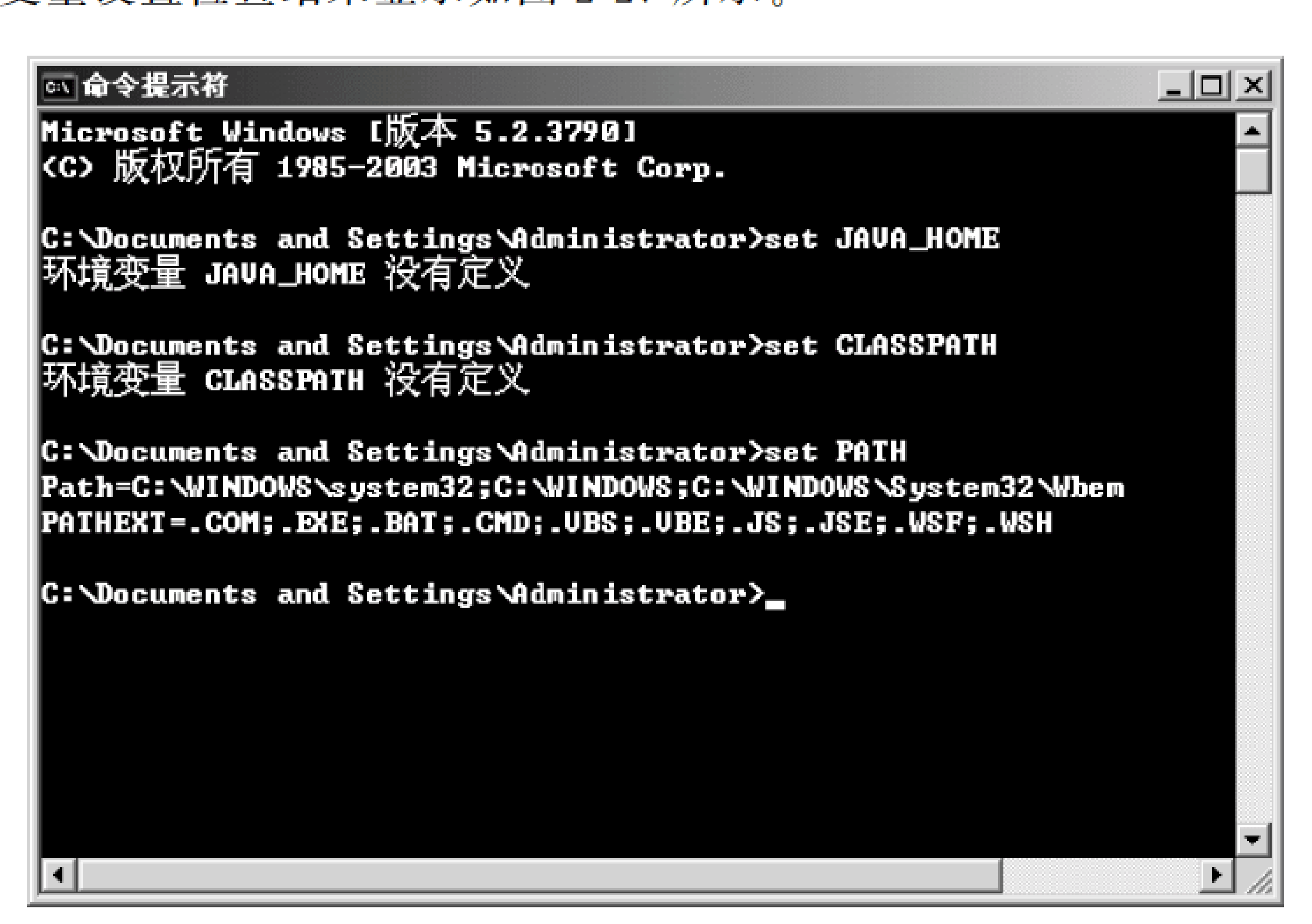



图 2-27 检查系统环境变量设置

如果安装程序没有设置 Java 运行环境需要的环境变量，应该进行手工设置。根据上述的 JDK 安装路径，设置内容如下：

```
set JAVA_HOME = C:\Java\jdk1.6.0_03
set CLASSPATH = .;%JAVA_HOME%\lib(注意:".;"一定不能少,它代表当前路径)
PATH = %PATH%;%JAVA_HOME%\bin;%JAVA_HOME%\jre\bin
```

各环境变量功能如下：

JAVA-HOME 表示 Java 的安装目录，在其他环境变量中使用。

CLASSPATH 定义 Javac 搜索类的路径，它记录 Java 编译器和解释器所需要的类所在的路径。即使是用户自己创建的类，也应该添加到 CLASSPATH 中，这样比较麻烦，所以在 CLASSPATH 中添加了一个当前目录(即“.；”)。这样，当转到用户所在的目录的时候，由于 javac 编译生成的用户类保存在当前路径，必须把当前路径加到 CLASSPATH 中，这样 Java 解释器才能够找到用户的类。有时候，会看到 CLASSPATH 中包含一个.jar 等压

缩的 class 文件[①]，把它加入到 CLASSPATH 中，Java 环境可以读取该文件。

PATH 变量是系统搜索可执行程序的路径，其中，Java 编译器(javac. exe)保存在%JAVA_HOME%\bin 中，Java 解释器(java. exe)保存在%JAVA_HOME%\jre\bin 中，要在任何路径下使用 javac. exe 和 java. exe，则必须将上述路径定义在操作系统的 Path 环境变量中。

要设置上述环境变量，需要通过控制面板中的“系统”程序来完成，具体步骤如下：

在 Windows“控制面板”中，双击“系统”图标，打开“系统属性”对话框，选择“高级”选项卡，如图 2-28 所示。在“高级”选项卡中，单击“环境变量”按钮，打开“环境变量”对话框，如图 2-29 所示。

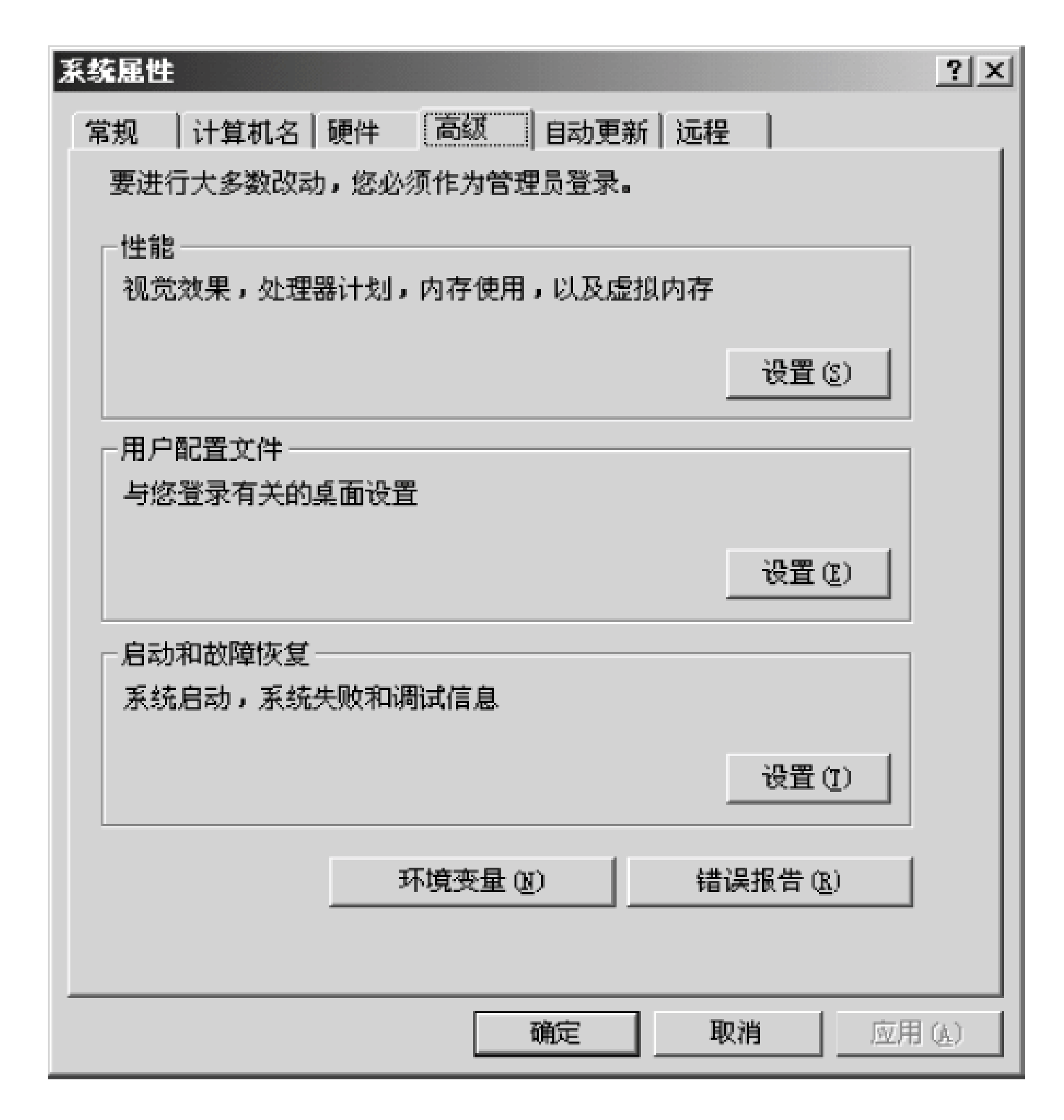



图 2-28 “系统属性”对话框




图 2-29 环境变量界面

在环境变量窗口的“系统变量”区域，可以新建环境变量，或者对已经存在的环境变量进行修改。

1) 设置 JAVA_HOME 环境变量

在“系统变量”区域，单击“新建”按钮，打开“新建系统变量”对话框，输入要新建的系统变量以及变量值，如图 2-30 所示。

输入完成后，单击“确定”按钮。

2) 设置 CLASSPATH 环境变量

用同样的方法，新建系统变量 CLASSPATH，如图 2-31 所示。




图 2-30 新建系统变量 JAVA_HOME

① jar 的全称是 JavaTM Archive(JAR)file，是 Java 存档文件，主要压缩存储 Java 的 class 文件。jar 是 JavaJDK 中的命令，可以在 DOS 提示符下使用 jar-help 命令显示 jar 的使用方法。

3）更新 PATH 路径设置

在“环境变量”窗口“系统变量”区域(参见图 2-29)，选择 Path 环境变量，单击“编辑”按钮，在原有 Path 基础上，增加“；%JAVA_HOME%\bin；%JAVA_HOME%\jre\bin”，如图 2-32 所示。

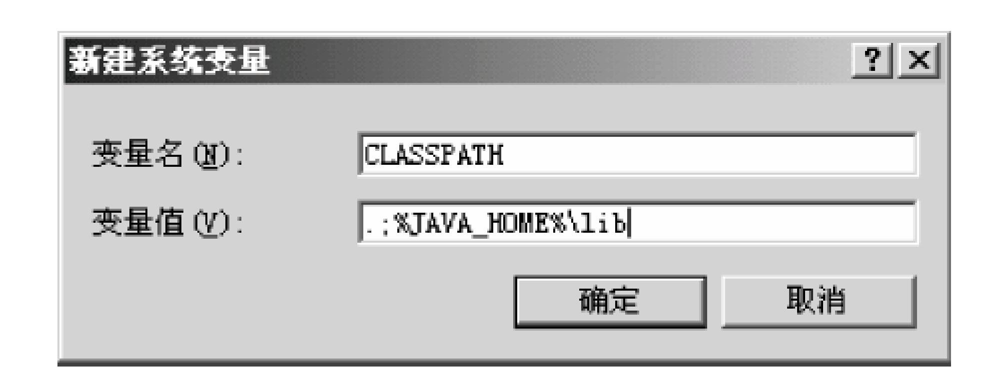



图 2-31　新建系统变量 CLASSPATH

图 2-32　更新系统变量 Path

#### 4. 测试 Java 运行环境

设置完成后，重新启动计算机，使上述设置生效。然后在 DOS 提示符下，依次输入下述命令来检查环境变量的设置情况：

```
c:\> echo %java_home%
c:\> echo %classpath%
c:\> echo %path%
```

也可以通过 set<环境变量名>命令来检验上述设置。如果设置正确，可以输入下列命令检查 Java 的运行是否正常。

```
c:\> java -version
c:\> javac
```

输入上述命令后，运行结果如图 2-33 所示。

如果能运行 Java 编译命令 javac 表明 Java 的环境变量设置就没问题了，接下来可以用一个简单的 Java 程序来测试 J2SDK 的安装。代码如下：

```
public class Test
{
 public static void main(String args[]) {
        System.out.println("Hello,My Java program ");
 }
}
```

创建文件夹 D:\MyJava，将上述程序代码保存在该文件夹下，文件名为 Test.java(和类名一致，包括大小写)。然后打开 DOS 命令提示符窗口，转到 Test.java 所在目录 D:\MyJava，然后输入下面的命令：

```
javac Test.java
java Test
```

如果显示“Hello，My Java program”，表明 Java 环境安装成功，用 Dir 命令可看到生成一个 Test.class 的文件。否则需要仔细检查环境配置情况。

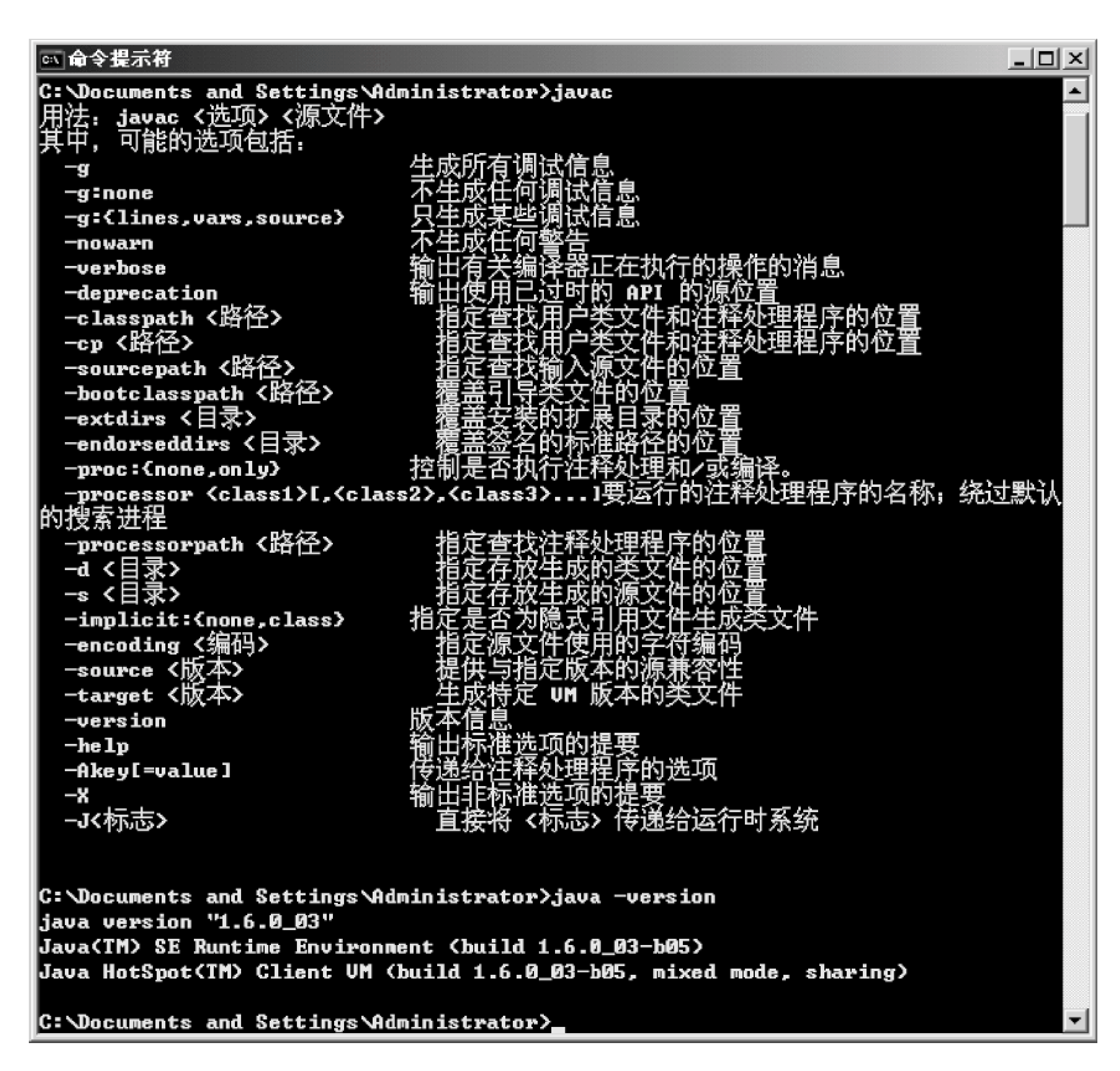



图 2-33　检验 Java 运行情况

### 2.5.5　Tomcat 的安装和配置

首先登录 Tomcat 官方网站 http://tomcat.apache.org/，在 Download 区域单击“Tomcat 6.x”超链接，显示 Tomcat 6.x 的下载界面，在 Tomcat 6.0.14 的二进制代码发布(Binary Distributions)区域，单击“Windows Service Installer(pgp，md5)”超链接，即可下载 Tomcat 安装程序，文件名为 apache-tomcat-6.0.14.exe。

#### 1. 安装步骤

执行 Tomcat 安装程序 apache-tomcat-6.0.14.exe，启动安装向导，按照向导提示执行下面步骤：

第一，选择要安装的 Tomcat 组件，如图 2-34 所示。

在安装类型下拉列表中，选择完全安装(Full)，Tomcat 将作为 Windows 服务器的服务直接启动。

第二，选择安装的物理路径，默认路径为：C:\Program Files\Apache Software Foundation\Tomcat 6.0。如图 2-35 所示。

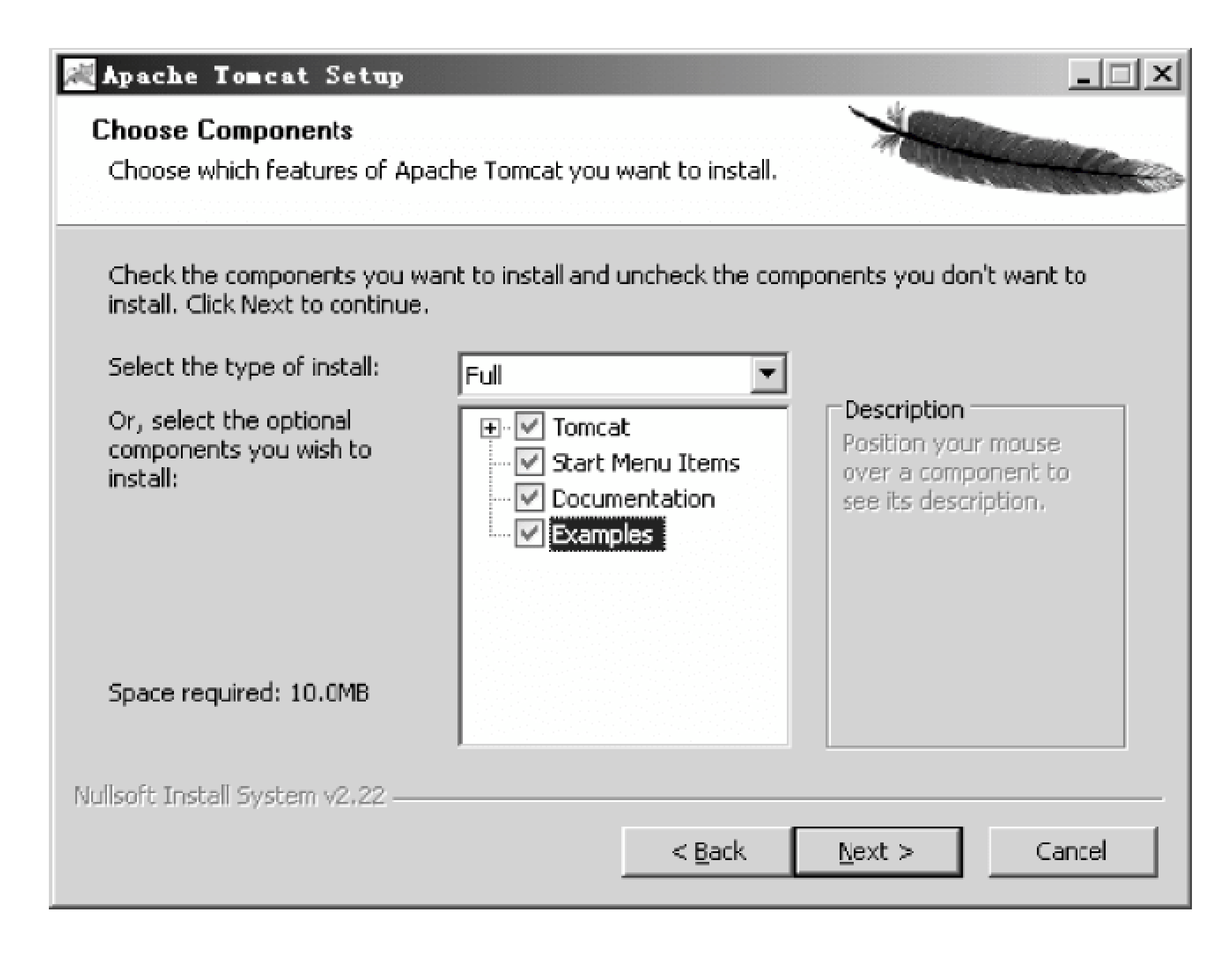



图 2-34 Tomcat 安装向导界面

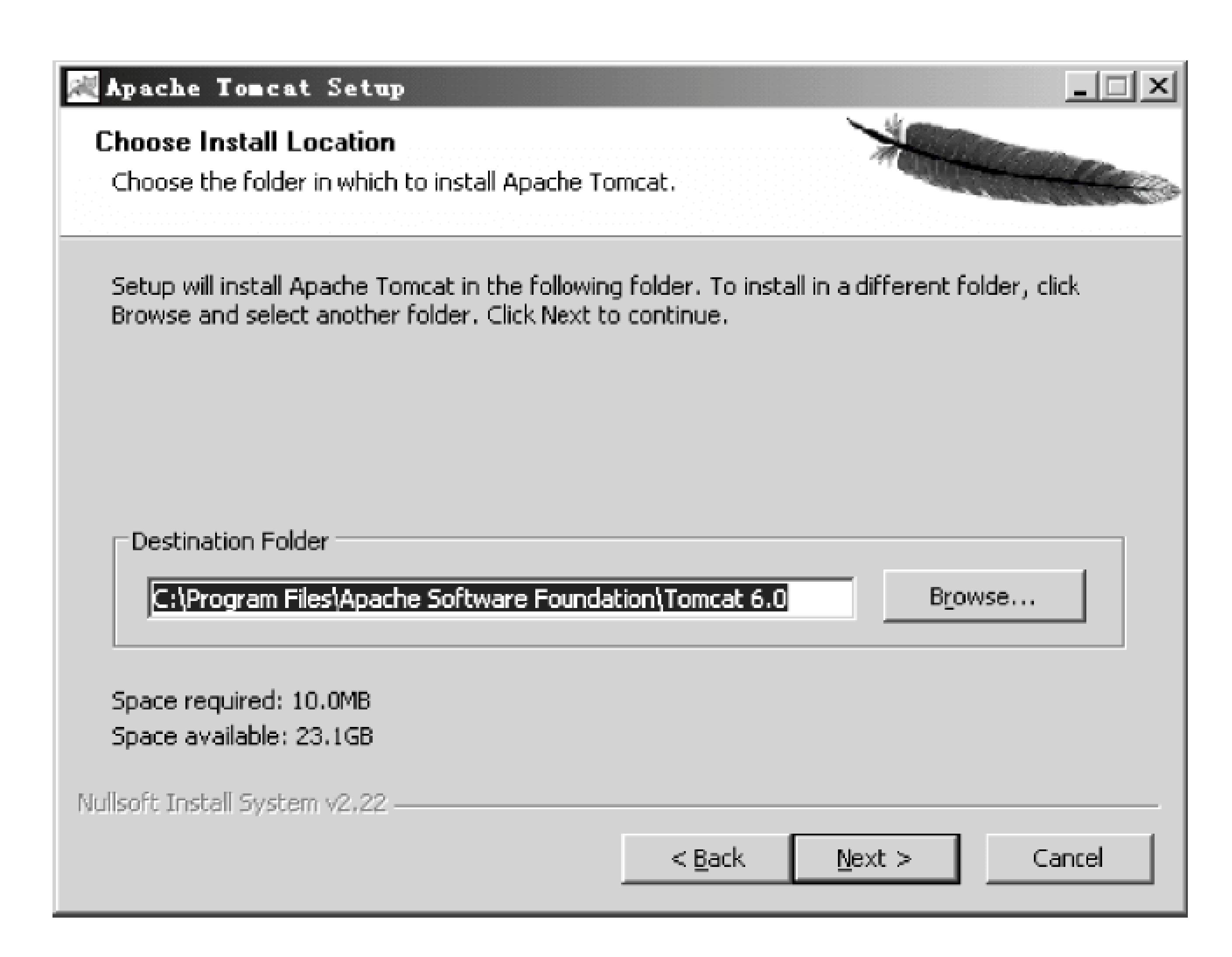



图 2-35 选择 Tomcat 安装路径

为下一步配置环境变量方便,我们修改安装路径为 C:\Tomcat 6.0。

第三,进行 Tomcat 的基本配置,包括 HTTP 端口,Tomcat 的默认值为 8080,可以修改为 80;管理员的登录名和密码,默认登录名为 admin,密码可以为空。如图 2-36 所示。

第四,选择安装 Java Virtual Machine 的物理路径。如果已经成功配置完毕 JDK(含 JRE),此时,向导直接指向 J2SDK 中安装的 JRE 目录,例如:C:\java\jdk1.6.0_03。如图 2-37 所示。

最后单击 Install 按钮,开始安装,向导将把有关的文件复制到相关的目录下,并自动启动 Tomcat。Tomcat 安装完成后,在开始菜单的"程序"组中,将增加 Apache Tomcat 6 程序组。

第五,测试安装是否成功。打开 IE 浏览器,在地址栏中输入 http://127.0.0.1:8080/

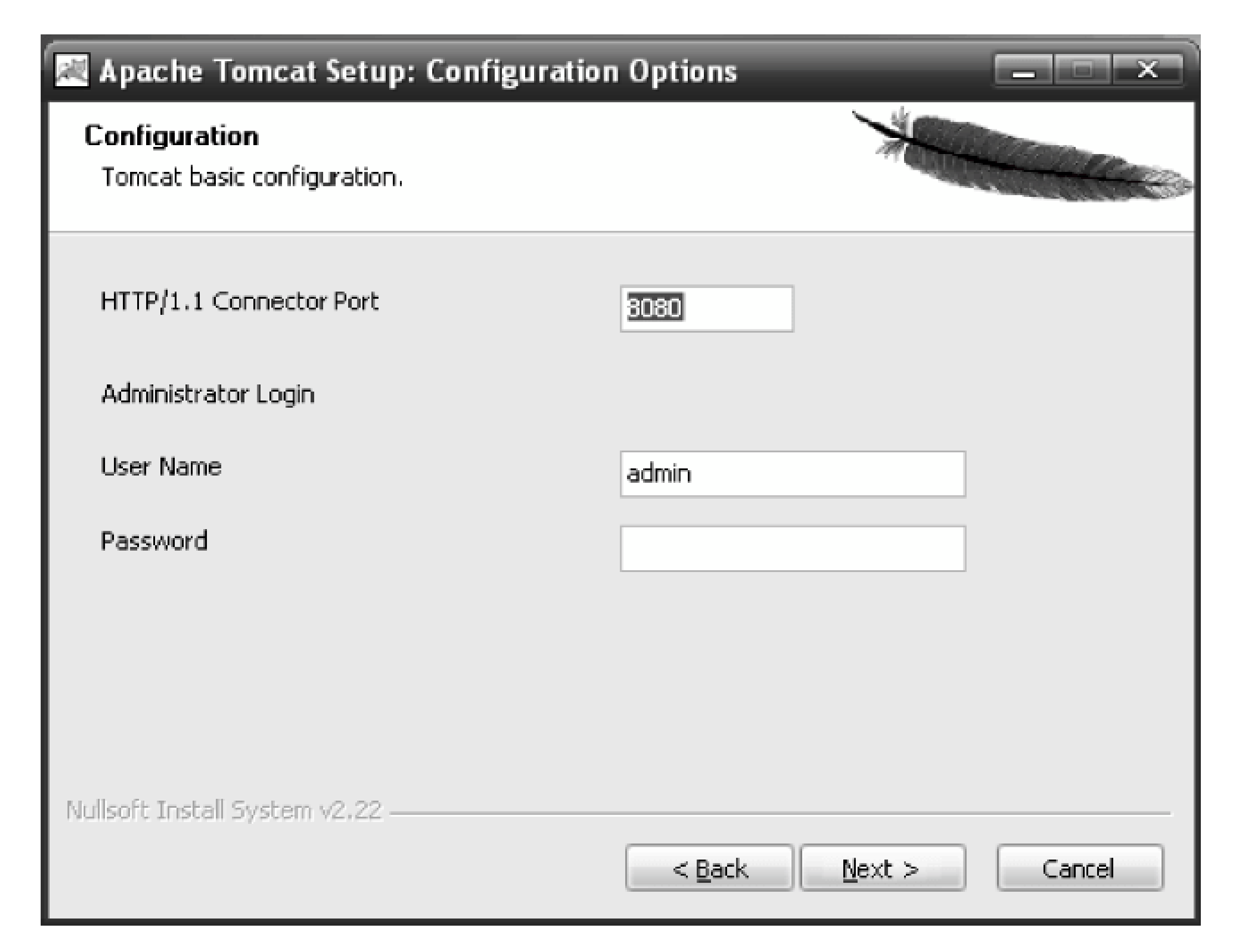



图 2-36 设置 Tomcat 服务端口号

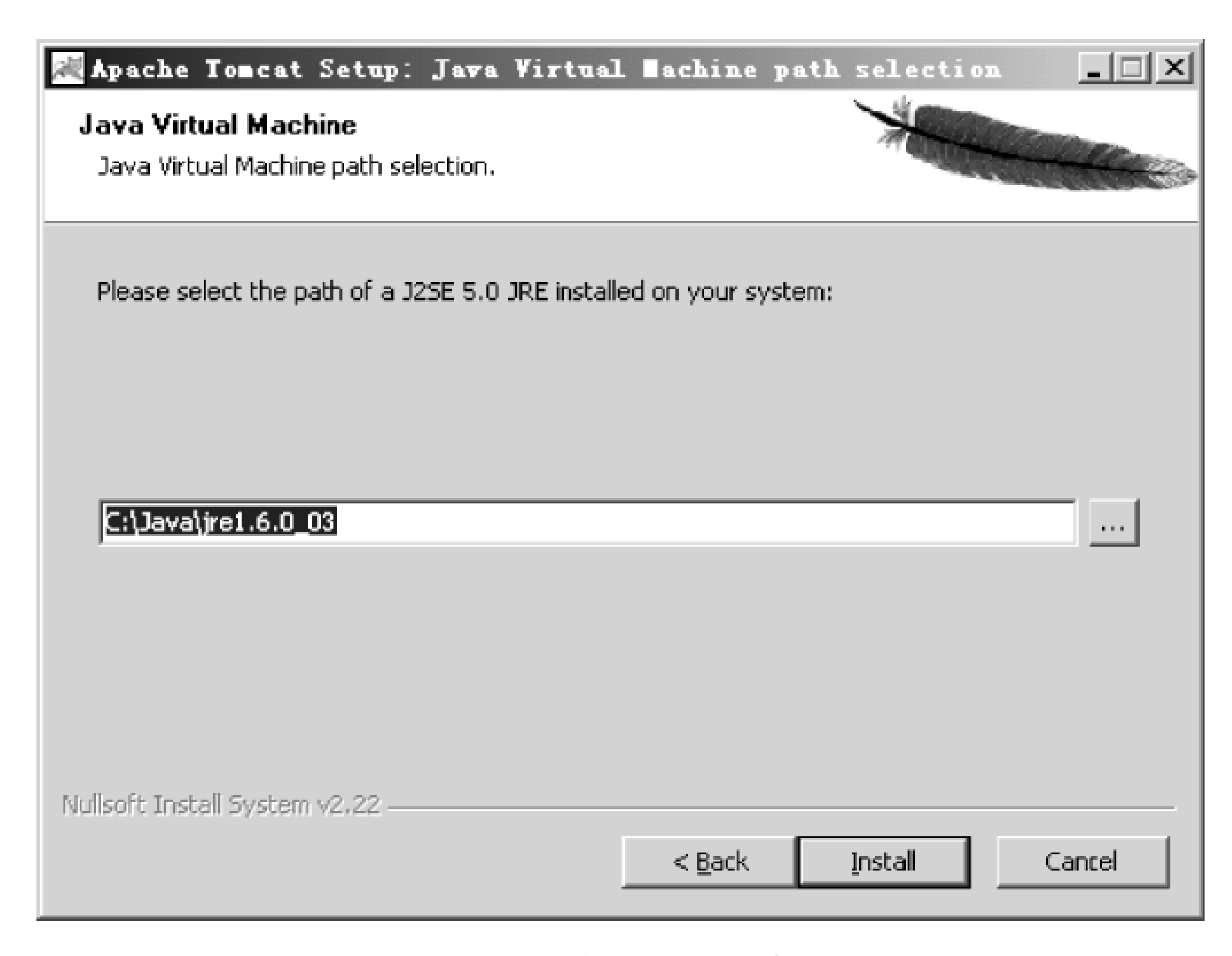



图 2-37 指向 JRE 路径

(或 http://localhost:8080/),如果出现如图 2-38 所示的界面,则表明 Tomcat 安装成功。

Tomcat 安装完成后,安装程序将建立相应的目录,所建立的目录结构如图 2-39 所示。

不同的 Tomcat 版本,安装完成后的文件夹结构不同,Tomcat 6.0 的文件夹结构比 Tomcat 5.5 简单,各文件夹及其功能说明如下:

- bin 目录下主要存放 Windows 平台上启动和关闭 Tomcat 的脚本。
- lib 目录存放 Tomcat 服务器以及所有 Web 应用都可以访问的 jar 文件。需要注意的是,为了在 Java 环境下能够正确编译 Servlet 文件,最好把 lib 目录中的 jsp-api.jar 和 servlet-api.jar 复制到 J2SDK 的安装目录的 lib 子目录(即 C:\Java\jdk1.6.0_03\lib)内,同时,需要在 CLASSPATH 环境变量中也增加这两个.jar 文件(即在原变量后面输入"; c:\java\jdk1.6.0_03\lib\jsp-api.jar; c:\java\jdk1.3.0_03\lib\servlet-api.jar")。

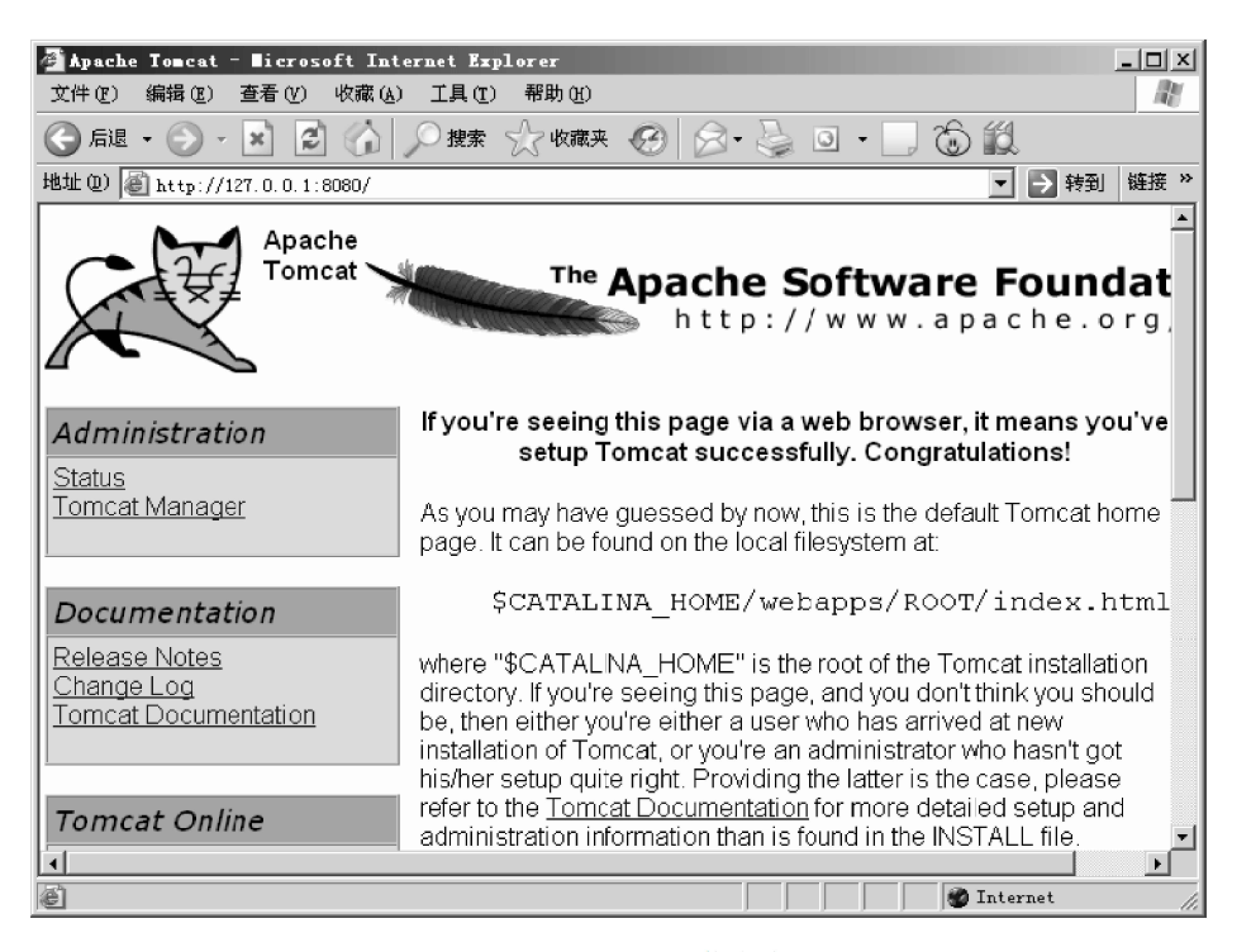



图 2-38 Tomcat 安装成功




图 2-39 Tomcat 安装目录结构

- conf 目录存放 Tomcat 服务的配置信息文件，其中最重要的是 server. xml 和 web. xml。server. xml 是 Tomcat 的主要配置文件，可以在其中配置 Web 服务的端口、会话过期时间、虚拟主机等。web. xml 为不同的 Tomcat 配置的 Web 应用设置默认值。另外，在其/Catalina/localhost 子目录下还可以设置网站虚拟目录和根路径信息等。
- logs 目录存放 Tomcat 执行时的 Log(日志)文件。
- temp 目录存放 Tomcat 运行的一些临时文件。
- webapps 目录存放 Tomcat 服务器自带的两个 Web 应用——host-manager 应用和 manager 应用。ROOT 子目录下存放默认首页，即输入 http://127. 0. 0. 1:8080/后启动的页面。
- work 目录存放 JSP 文件在运行时被编译成的二进制文件(Servlet)。在 localhost 文件夹下包含了多个子文件夹，其中第一个文件夹"_"对应 Web 服务的根，Tomcat 执行主 Web 应用的 JSP 页面时生成的临时文件将存储在"C:\Tomcat 6. 0\work\Catalina\localhost_"文件夹中。其他文件夹分别对应虚拟目录，每建立一个虚拟目录，在 localhost 文件夹中将创建一个同名的子文件夹。用户可以删除整个 localhost 子文件夹，来删除所有的临时文件。

有时修改页面内容后，仍然显示修改以前的内容，这时可以尝试把 work/Catalina/localhost 目录中所有内容删除，如果删除时出现无法删除提示，需要关闭 Tomcat，然后再删除。然后重启 Tomcat 即可正确显示我们修改后预期的内容。

在 Tomcat 6 以前的 Tomcat 5. 5 中，有三个不同的 lib 目录，分别存储在/server、/common 和/shared 目录下，这些 lib 目录都可以存放 jar 文件。那么它们有哪些区别呢？区别主要在于：

- /server/lib 目录下的 jar 文件只可被 Tomcat 服务器访问。
- /common/lib 目录下的 jar 文件可以被 Tomcat 服务器和所有 Web 应用访问。
- /shared/lib 目录下的 jar 文件可被所有 Web 应用访问，而不能被 Tomcat 服务器访问。

在用户自己的站点中，WEB-INF 目录下也可以建立 lib 子目录，在 lib 子目录下也可以存放各种 jar 文件，但这些 jar 文件只能被当前 Web 应用访问。

接下来即可进行 Tomcat 的配置，分成四个方面：根据上述目录结构进行相应的环境变量设置，配置 Tomcat 服务端口，设置 Tomcat 服务根目录，建立虚拟目录。

### 2. 配置 Tomcat 环境变量

Tomcat 为 JSP 的容器，要在 Windows 下运行 JSP，需要安装 Java 开发环境，同时需要一些特殊的环境设置，包括以下四个系统环境变量，具体内容应根据安装路径设置。

1) 添加 Tomcat 主目录环境变量

```
set TOMCAT_HOME = C:\Tomcat 6.0
```

2) 添加 CATALINA_HOME 环境变量

```
set CATALINA_HOME = C:\Tomcat 6.0
```

3）更新 CLASSPATH 环境变量

```
CLASSPATH = . ; % JAVA_HOME % \lib; % TOMCAT_HOME % \lib
```

4）更新 PATH 环境变量

```
PATH = % PATH % ; % TOMCAT_HOME % ; % TOMCAT_HOME % \bin
```

上述环境变量的配置和 Java 环境变量的配置方法相同。设置完成后，重新启动计算机，使设置生效，然后再启动 Tomcat。

需要特别注意的是，如果该步骤的环境变量配置不对或者 server. xml 文件配置不对（见下面的介绍），Tomcat 将无法启动。另外，如果 Web 应用中只是一般的 HTML 文件，不配置环境变量，网站也可以浏览，因此，Tomcat 启动后，并不意味着所有的需要运行用户 Web 的设置都完成或正确。

在实际应用中，一般需要更改三个基本配置：修改服务端口、修改网站的根路径和建立虚拟目录。在以前的 Tomcat 版本中，这些配置比较复杂。在 Tomcat 6 中，这些配置都是通过 Tomcat 主配置文件 conf/server. xml 完成的。

### 3. 修改服务端口

在 Tomcat 的安装过程中，可以设置 Tomcat 服务端口，默认值为 8080。安装完成后，如果需要修改服务端口，可通过 Tomcat 主目录下的 conf 目录中的 server. xml 文件完成。不同的 Tomcat 版本，主配置文件 Server. xml 的内容不同。

对于 Tomcat 6. 0. x，利用 UltraEdit 或其他文本编辑器打开 C:\Tomcat 6. 0\conf\目录下的 Server. xml 文件，定位元素＜Connector port＝"8080"＞，可以看到 Tomcat 服务的设置端口为 8080，如图 2-40 所示。

```
    <!-- A "Connector" represents an endpoint by which requests are received
         and responses are returned. Documentation at :
         Java HTTP Connector: /docs/config/http.html (blocking & non-blocking)
         Java AJP  Connector: /docs/config/ajp.html
         APR (HTTP/AJP) Connector: /docs/apr.html
         Define a non-SSL HTTP/1.1 Connector on port 8080
    -->
    <Connector port="8080" protocol="HTTP/1.1"
               connectionTimeout="20000"
               redirectPort="8443" />
```

图 2-40　Tomcat 的服务端口信息

修改 Web 服务端口为 http 的默认端口 80。注意，如果是在 Windows 平台中，并且安装了 IIS，则修改的端口号不要和 IIS 中的 Web 服务冲突。修改完毕后，保存该文件，然后重启 Tomcat 服务器，这样 Tomcat 就在新的端口提供服务了。

### 4. 修改网站根路径

不同的 Tomcat 版本，设置 Web 应用根的方法也不相同。在 Tomcat 5. 5. x 中，修改网站根路径的方法有两种：一种是修改 C:\Tomcat 5. 5\conf\目录下的 Tomcat 主配置文件 server. xml；一种是建立 ROOT. xml 文件。在 Tomcat 6. 0. x 中，设置 Tomcat 根的方法非常简单，只需要修改 Tomcat 主配置文件 conf/server. xml 即可。

用记事本打开 Tomcat 主配置文件 Server. xml，定位到文档尾部的<Host>元素，添加一个上下文元素（<Context>），来设置 Tomcat 的根。例如，如果将 D:\GSL3.0 设置为 Tomcat 的根，设置如图 2-41 所示。

```
server.xml - 记事本
文件(F)  编辑(E)  格式(O)  查看(V)  帮助(H)
      <!-- Define the default virtual host
           Note: XML Schema validation will not work with Xerces 2.2.
       -->
      <Host name="localhost"  appBase="webapps"
            unpackWARs="true" autoDeploy="true"
            xmlValidation="false" xmlNamespaceAware="false">

        <!-- SingleSignOn valve, share authentication between web applications
             Documentation at: /docs/config/valve.html -->
        <!--
        <Valve className="org.apache.catalina.authenticator.SingleSignOn" />
        -->

        <!-- Access log processes all example.
             Documentation at: /docs/config/valve.html -->
        <!--
        <Valve className="org.apache.catalina.valves.AccessLogValve" directory="l
               prefix="localhost_access_log." suffix=".txt" pattern="common" reso
        -->
      <Context path="" docBase= "d:\GSL3.0" />
      </Host>
    </Engine>
  </Service>
</Server>
```

图 2-41　设置 Tomcat 服务的根

需要特别注意的是，Tomcat 区分大小写，<Context>元素的第一个字母一定为大写，且文件夹名称大小写也必须和实际一致。修改完毕后，在任务栏上停止 Tomcat，再重新启动，打开浏览器，将运行根中的 index. jsp 程序页面。

如果还要建立虚拟目录，只需要再增加不同的<Context>元素即可，详细介绍见 2.5.7 节的内容。在 Server. xml 中，可以设置多个不同的虚拟目录。

### 5. 设置 Web 应用首页

在 Windows IIS 中，我们可以设置一个 Web 站点的首页（即：登录一个站点，在不指定下载文件时默认的下载文档，一般是存储在站点主目录下的 index 文件）。在 Tomcat 中，如何设置站点首页呢？

在 Tomcat 中，站点首页是通过 web. xml 文件完成的，web. xml 文件又称为站点配置文件。在每一个 Web 应用中，往往在主目录下包含一个 WEB-INF 子目录，其中存储了该站点的配置文件 web. xml。此外，在 Tomcat 的 conf 文件夹下也包含一个 web. xml 文件，内容如下：

```
    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
        <welcome-file>index.htm</welcome-file>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>

</web-app>
```

Tomcat 的 conf/web.xml 文件是对所有 Web 应用的一个公共配置。对于一个具体的 Web 应用，如果包含自己的 WEB-INF/web.xml 文件，当两个配置冲突时，则自己的 web.xml 配置将覆盖 conf/web.xml 中的设置。一般情况下，只需要修改 conf/web.xml 配置文件即可，不需要单独设置每一个应用的 WEB-INF/web.xml 文件。

### 2.5.6 建立并部署 Web 应用

在默认情况下，Tomcat 指向一个默认的 Web 应用(C:\Program Files\Apache Software Foundation\Tomcat 5.5\webapps\ROOT)，在 webapps 文件夹下，还包含其他的几个 Web 应用，如 jsp-examples、servelets-examples 等。

下面介绍在 Tomcat 下新建 Web 应用的方法和步骤。

#### 1. 规划 Web 应用目录结构

对于一个 Web 应用，包含了大量的网页文件，为了更好地管理和维护，应该按照一定的规则组织文件。常用的方法是按照 Web 站点功能建立文件夹，分别存储相应的页面文件。图 2-42 是我们根据一个常用的 Web 应用规划的文件夹结构。

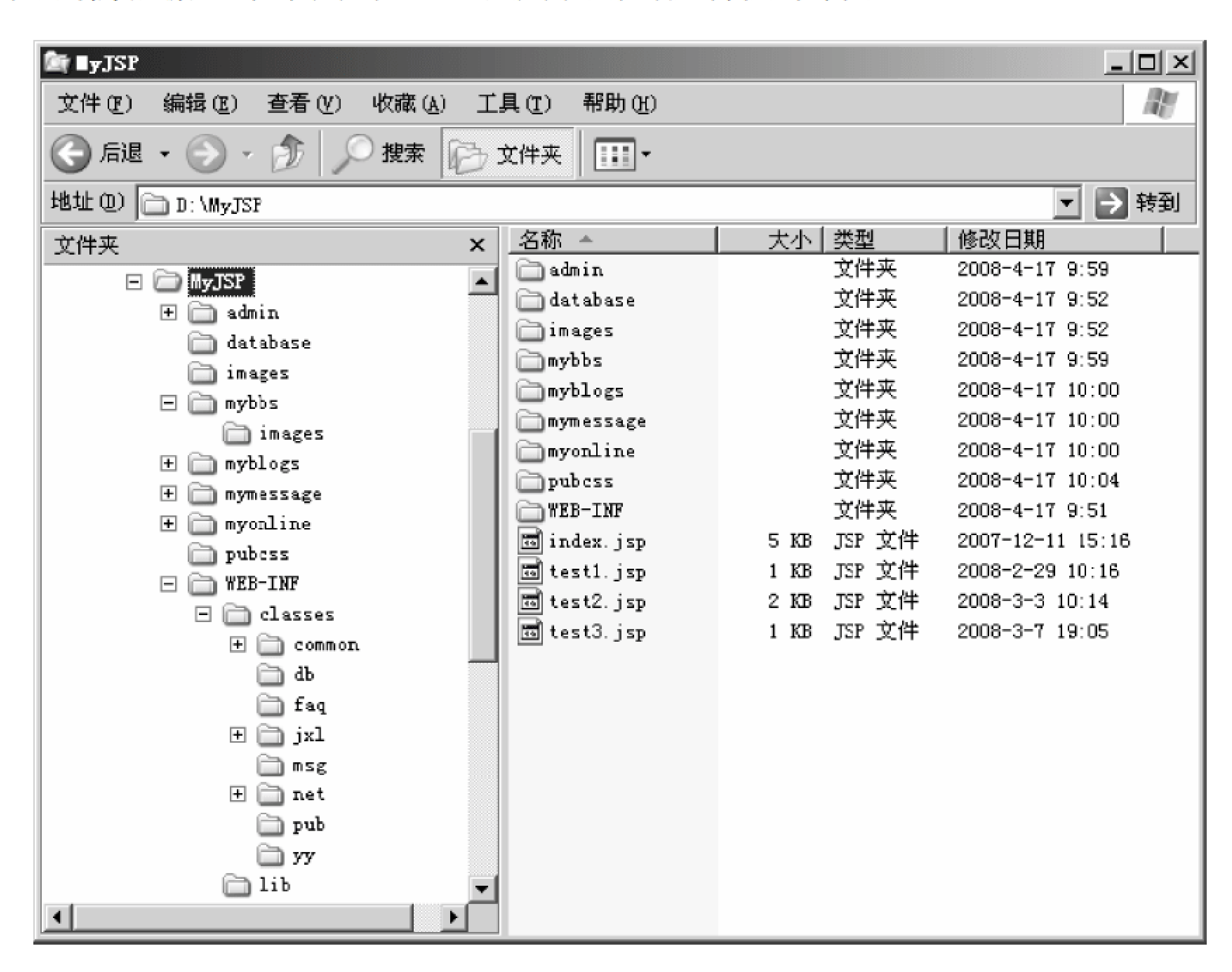



图 2-42 Web 应用目录结构示例

将每一类功能相关的页面、图片组织到一个文件夹中，例如：mybbs、myblogs、myonline 等，在这些文件夹中，还可以定义子文件夹，例如定义 images 文件夹，存储所用到的图片。在站点主目录下，通常还可以定义 database 文件夹，存储站点数据库文件；定义 pubcss 文件夹，存储用户定义的样式表。

当然，还有一个 WEB-INF 文件夹，存储 Web 应用的配置文件 web. xml 以及定义 classes 和 lib 两个子文件夹，存储 Web 中用户定义的类。用户定义的大量的 JavaBean 都是存储在 WEB-INF/classes 文件夹中的，里面通常还定义不同的包，即子文件夹。

在站点主目录中，包含了 Web 应用的首页文件 index. jsp，也可以包含一些其他的常用文件，这些文件通常是公用的，不便于保存到一个具体的功能文件夹中。

### 2. WEB-INF 目录

在 Tomcat 中，每一个 Web 应用，主目录下往往都包含一个 WEB-INF 目录，用于放置一些配置文件与不希望外部程序访问的隐私文件，在网络上是不允许访问该文件夹的。在 WEB-INF 目录下有一个 Web 应用部署文件 web. xml，对当前应用程序进行相关设置，如设置 Web 应用的默认首页文件等。

在 WEB-INF 目录下还可以建立 classes 和 lib 子目录。classes 目录用于放置 Web 应用程序所需调用的类，如 JavaBean。在运行过程中，Tomcat 类装载器先装载 classes 目录下的类，再装载 lib 目录下的类。如果两个目录下存在同名的类，classes 目录下的类具有优先权。lib 目录主要是放置需要引入的 jar 文件，应用程序导入的包先从这里开始寻找，其次到容器的全局路径 $TOMCAT_HOME/lib 下寻找。

### 3. Web 应用配置文件 web.xml

对 Web 应用的配置是通过 Web 应用配置文件 web. xml 实现的，类似于 Windows IIS 中的站点属性对话框的配置。在 Tomcat/conf 下包含一个 Web 应用配置文件 web. xml，它是所有 Web 应用的公共配置文件。此外，在每一个 Web 应用中，在主目录下的 WEB-INF 子目录中，都包含一个 web. xml 文件，它是该 Web 应用的部署文件。当两个配置中的项目冲突时，则自己的 web. xml 配置将覆盖 conf/web. xml 中的设置。

在 web. xml 配置文件中，根元素是<web-app>，其中定义了站点的各种配置，主要包括以下几个方面：

(1) 网站名称和说明。包括三个 XML 元素，分别是：<description>、<display-name>、<icon>，用于设置站点的描述、显示名称和图标。例如，Tomcat 自带的 manger 应用的 web. xml 中的站点说明 XML 元素内容如下：

```
<display-name>Tomcat Manager Application</display-name>
<description>
  A scriptable management web application for the Tomcat Web Server;
  Manager lets you view,load/unload/etc particular web applications.
</description>
```

(2) Servlet 的名称和映射。servlet-mapping 元素包含两个子元素：servlet-name 和 url-pattern，用来定义 Servlet 所对应的 URL。

(3) Session 的设定。session-config 包含一个子元素 session-timeout，用于定义 Web 站点中的 session 参数。例如，设定会话时间为 20 分钟，对应的 XML 元素内容为：

```
<session-config>
      <session-timeout>20</session-timeout>
</session-config>
```

(4) mime 映射。mime-mapping 包含两个子元素：extension 和 mime-type，定义某一个扩展名和某一 MIME Type 映射。例如，要在 Tomcat 中打开 Excel 文件，需要在 web. xml 中作如下设置：

```
<mime-mapping>
    <extension>xls</extension>
    <mime-type>application/vnd.ms-excel</mime-type>
</mime-mapping>
<mime-mapping>
    <extension>csv</extension>
    <mime-type>application/vnd.ms-excel</mime-type>
</mime-mapping>
```

(5) 错误处理。error-page 元素包含三个子元素：error-code、exception-type 和 location，将错误代码(Error Code)或异常(Exception)的种类对应到 Web 站点的相应页面。例如：

```
<error-page>
    <error-code>404</error-code>
    <location>/error404.jsp</location>
</error-page>
```

(6) 默认首页设置。对应的元素声明一般形式为：

```
<welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
</welcome-file-list>
```

一般情况下，只需要配置 Tomcat 的 conf/web. xml 公共配置文件即可，不需要为每一个 Web 应用配置其 WEB-INF/web. xml 文件。

#### 4. 修改 Tomcat 配置

在测试我们的 Web 应用以前，需要对 Tomcat 做相应的设置，使得 Tomcat 指向用户的 Web 应用(例如 D:\MyJSP)，修改如下：

(1) 修改 Tomcat 主配置文件\Tomcat 6.0\conf\server. xml，设置 Web 服务的端口号为 80，同时，修改默认 Tomcat 服务的根，在 server. xml 的尾部，添加下列元素(参见图 2-40)：

```
<Context path = "" docBase = "d:\MyJSP" />
```

(2) 设置站点首页，可以修改 Tomcat 配置文件\Tomcat 6.0\conf\web. xml，设置 Web 应用的一些常用配置，默认首页为 index. jsp，无须修改。

需要注意的是，如果已经启动了 Apache Server，首先应该在 Windows 的开始菜单中，在程序组中找到“Apache HTTP Server ”程序组，执行 Stop 命令，停止 Apache Server。

#### 5. 测试新的 Web 应用

当上述修改完毕后，在任务栏中右击 Tomcat 图标，选择 Shutdown：Tomcat 命令，关闭 Tomcat。然后在“开始”菜单中重新启动 Tomcat，尝试运行用户 Web 应用。

在站点根下，建立一个简单的站点首页文件 index.jsp，代码如下：

```
<%@ page contentType="text/html;charset=gb2312"%>
<html>
<head>
    <title>Hello,JSP</title>
</head>
<body>
    <% out.println("你好,JSP!"); %><br>
    现在的时间是：<% = new java.util.Date() %>
</body>
</html>
```

打开 IE 浏览器，输入 http://127.0.0.1/，显示如图 2-43 所示。

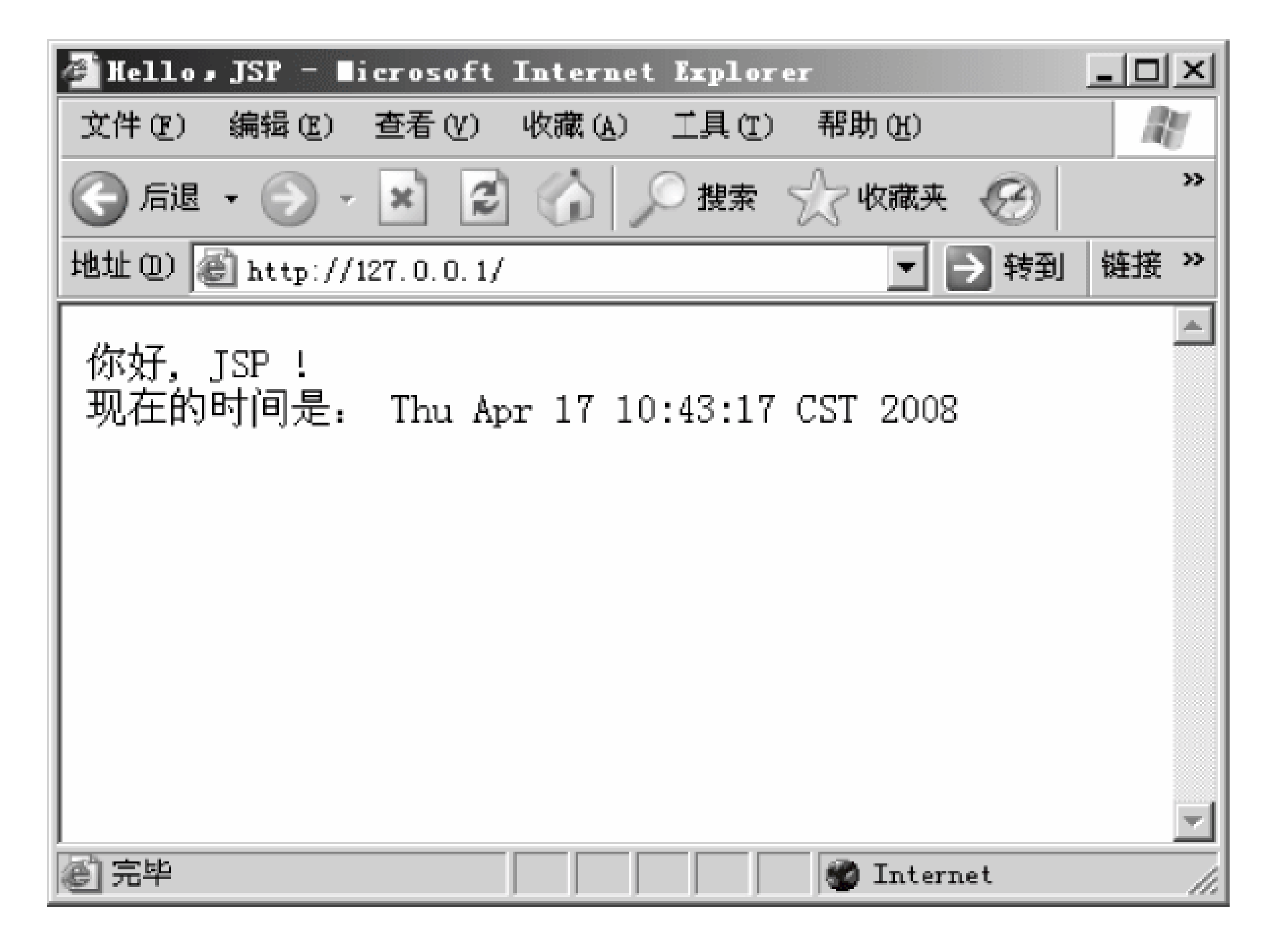



图 2-43 第一个 JSP Web 应用首页

表明 Tomcat 已经运行了用户的 Web 应用 D:\MyJSP 目录下的首页文件 index.jsp。用户可以在主目录下创建其他的 JSP 文件，在浏览器的地址栏内输入：http://127.0.0.1/文件名(包含扩展名)即可执行相应的 JSP 文件了。

如果 JSP 文件中含有 Java 脚本程序，必须要保证 Tomcat 和 J2SDK 的环境变量设置正确，否则 Web 应用将不能运行。如果 Web 页是.htm 文件，运行与环境变量的配置无关。

在应用中，如果遇到不能打开 Web 应用首页文件 index.jsp，应检查该 Web 应用主目录下的 WEB-INF\web.xml 配置文件，同时检查 Tomcat 下的公共配置文件 conf\web.xml，确认两者是否配置一致。如果修改了某个 JSP 页面，但重新运行仍显示原先内容，此时需要删除 Tomcat\work\Catlinia\localhost 中的所有临时文件，也可以直接删除 localhost 子文件夹。

## 2.5.7 使用虚拟目录

用户如果希望使用虚拟目录，例如，在不改变站点主目录的情况下，需要建立新的 Web 应用(对应站点主目录外的新的文件夹)，访问该 Web 应用需要使用虚拟目录，在浏览器地

址栏输入：http://127.0.0.1/虚拟目录/文件名。

在 Tomcat 6 中使用虚拟目录非常简单，只需要修改 Tomcat 主配置文件 conf\server.xml，在尾部增加一个新的<Context>元素即可。例如，建立一个到 D:/haosite 的虚拟目录，在 server.xml 中，在根目录设置的后面(参见图 2-41)增加下述内容：

```
<Context path = "/hao" docBase = "d:\haosite" reloadable = "true" crossContext = "true"
         Debug = "0" workdir = "d:\haosite\work">
</Context>
```

其中，path="/hao"定义了根下的一个虚拟目录 hao，docBase="d:\haosite"为虚拟目录 hao 对应的物理路径。参数 reloadable 设置为 true，表明修改 Servlet 文件、JSP 文件后，不用重启 Tomcat 即可生效。

保存 server.xml 文件，然后重启 Tomcat 服务器，可以在地址栏中通过虚拟目录访问 D:/haosite 中的网页文件了，例如：http://127.0.0.1/hao/index_hao.jsp。

此时，在 Tomcat 的临时文件夹 C:\Tomcat 6.0\work\Catalina\localhost 中，自动创建一个与虚拟目录同名的临时文件夹 hao，存储该虚拟目录生成的临时文件。

### 2.5.8 Apache 和 Tomcat 的关系

通过以上的介绍，可以看出只用 Tomcat 也能够建立和运行一个 Web 站点，那么 Apache 和 Tomcat 是一种什么关系呢？是不是两者必须一起使用呢？

实际上 Apache 和 Tomcat 的作用是不一样的，Apache 主要是要实现虚拟主机，支持 PHP、站点性能、安全等方面时才需要。如果不是要用 Apache 实现以上功能，从开发的角度没必要用 Apache 和 Tomcat 配合，Tomcat 一个就可以完全应付了。也就是说，不需要安装 Apache 服务器，单独使用 Tomcat 即可运行 Web 应用。这是因为，Tomcat 内置了一个 Apache 的 HTTP 服务，但是它仅仅对 JSP 程序体现出比较好的执行效率和性能，对于静态页面的处理速度远不如 Apache。

可见，为了提高 Web 系统的整体性能，需要将 Apache 和 Tomcat 进行整合配置，具体的实现请读者参考其他书籍或从网上查找。

## 2.6 IIS 和 Tomcat 的整合

通过上面的介绍，我们知道 IIS 运行在 Windows 平台下，主要的开发工具是 ASP。Tomcat 主要是基于 Java 和 JSP 开发的。能否在 IIS 中使用 JSP 呢？也就是说通过 IIS 建立的 Web 站点，可以执行 JSP 文件，这就需要 IIS 和 Tomcat 的整合。IIS 和 Tomcat 的整合主要是通过 IIS 中 Web 站点属性对话框的“ISAPI 筛选器”选项卡进行相应的设置。

对于 IIS 和 Tomcat 的系统整合比较麻烦，网上有大量的文章介绍，比较完整的介绍请参考 http://www.hclab.com/hclabdata/list.asp? id=177 中的“IIS 和 Tomcat 整合解决方法”和 http://www.reynir.net/tomcat/tomcat_IIS_service_jk2.html 中的“Tomcat and IIS Installation process(jk2)”，并且有所需要的文件代码的附件。具体的整合步骤介绍在

此省略。

另外，对于在 IIS、Apache 和 Tomcat 建立的不同的 Web 应用，能同时运行吗？答案是肯定的。如果在一台计算机上建立了多个不同的 Web 应用，又分别采用的是 IIS 和 Tomcat，此时只需要给每个不同的 Web 应用不同的端口号即可。

同时，由于 Tomcat 建立的 Web 应用对文件的大小写是敏感的，我们还可以将两种应用融合起来。例如，在基于 Tomcat 的 Web 应用中通过超链接连接到 IIS 中的 Web 应用。方法是：在 Tomcat 的 Web 中增加一个指向 IIS 中 Web 应用的链接 http://IP 地址：端口号/，这里用的是 IIS 中建立的 Web 应用的首页。这样多个 Web 应用就可以在一台计算机上同时运行了。

## 2.7　Web 服务器的远程管理

随着互联网的发展，服务器的远程管理成为最主要的管理模式。在 Windows Server 2003 平台中，只要在服务器上进行相应的配置，就可以实现对服务器的远程管理。

### 2.7.1　使用终端服务和远程桌面

在 Windows Server 中，都提供了终端服务组件。因此，管理员可以在 Windows 服务器上安装 Windows 服务组件"终端服务"和"远程桌面 Web 连接"，然后管理员就可以使用 Windows XP 中的"远程桌面连接"（在"附件"、"通信"程序组中）来登录到 Web 服务器，实现对服务器的操作。

执行"附件"、"通信"程序组中的"远程桌面连接"，首先显示"远程桌面连接"对话框，如图 2-44 所示。




图 2-44　"远程桌面连接"对话框

输入要连接的计算机的 IP 地址，单击"连接"按钮，显示"Windows 登录"对话框，输入远程计算机上的一个本地账户和密码，即可登录到远程服务器，显示其桌面，接下来就如在本地一样对远程的计算机进行操作和管理了。

### 2.7.2　基于浏览器的服务器远程管理

在 Windows Server 平台的 IIS 的"万维网服务"组件中，包含了"远程管理(HTML)"子组件，安装该组件后，可以对 Windows 服务器进行远程管理。

具体设置如下：

(1) 在“计算机管理”控制台中，在“服务和应用程序”结点下，展开“Internet 信息服务”管理单元。

(2) 在需要远程管理的 Web 站点上右击鼠标，打开快捷菜单，执行“属性”命令，打开 Web 站点属性对话框。在 Web 站点选项卡中，记下该站点的 TCP 端口号。

(3) 在 Web 站点属性对话框中，选择“目录安全性”选项卡，在“IP 地址和域名限制”区域，单击“编辑”按钮，打开“IP 地址和域名限制”对话框，执行下列操作之一：

- 如果要允许所有计算机远程管理 IIS，单击“授权访问”单选钮。
- 单击“拒绝访问”单选钮，然后单击“添加”按钮，打开“授权访问”对话框，选择要授权访问的“单机”、“一组计算机”或者“域名”，按照系统提示进行操作。

当 Web 服务器上启用了基于浏览器的 Internet 服务管理器(HTML)后，就可以使用基于浏览器的 Internet 服务管理器了。

在浏览器地址栏输入：https://Web 服务器网址(域名或 IP 地址):8098/，回车，显示“连接到…”对话框，输入一个管理员权限的用户账户和密码，则打开“服务管理”站点，即通过 Web 接口远程维护 Windows Server 2003 服务器界面，如图 2-45 所示。




图 2-45 Web 接口远程维护 Windows Server 2003 服务器界面

通过 Web 接口，可以实现 Windows Server 2003 服务器的远程维护，包括：站点、Web 服务器、网络、用户等维护功能。

### 2.7.3 对网站的远程管理

如果要对 Windows 服务器平台中的网站进行远程管理，可以有很多方式，除了上述所讲的几种管理方式外，还可以利用 FTP 和 FrontPage 等进行远程管理，其中采用 FTP 管理网站具有更好的通用性，可以管理各种类型的网站；使用 FrontPage 只能管理 IIS 中的网站，如果是 Tomcat 中的网站，则不能通过 FrontPage 来管理。

使用 FTP 管理网站，需要对 Web 服务器进行如下配置：

(1) 在 Web 服务器计算机上搭建一个 FTP 站点，设置其主目录为要管理的网站主目录。

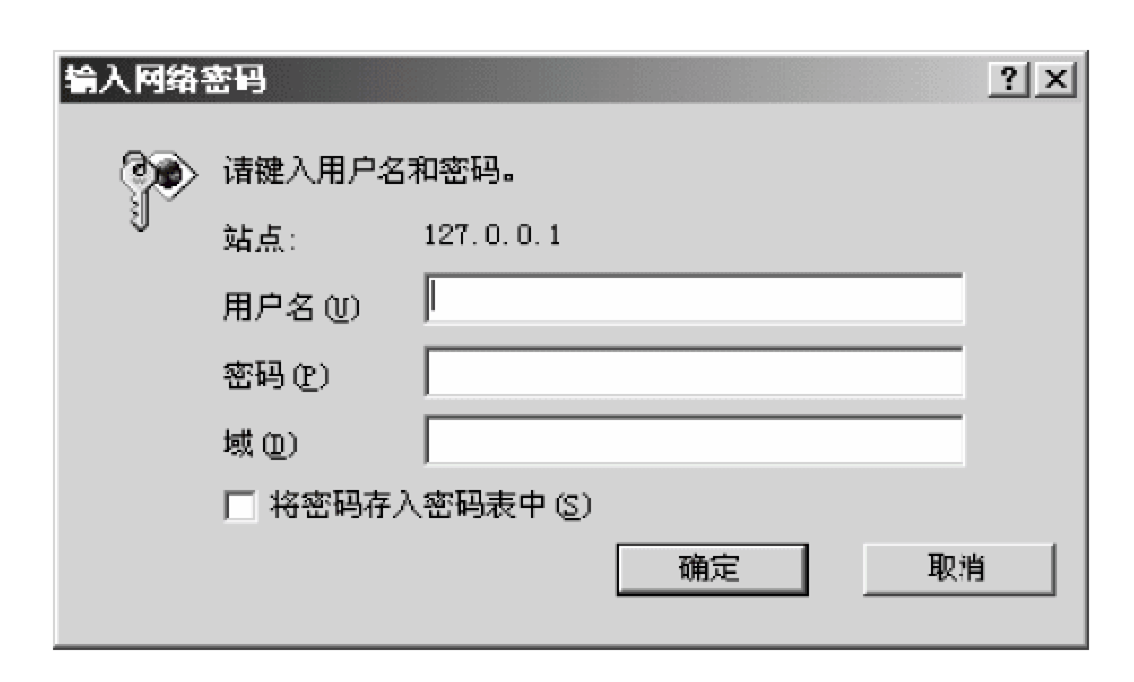
(2) 用户登录 FTP 站点，即可看到网站主目录，接下来就可以对网站进行维护操作了。例如：修改网页，首先将要修改的网页下载，修改完毕后再上传，覆盖原先的网页。

可见，该方式与网站是 IIS 的还是 Tomcat 的无关，也不需要对要管理的网站的属性进行特殊的设置，只需要 FTP 指向站点主目录即可。

对网站的远程维护还可以通过 FrontPage、Dreamweaver 等工具来完成，这些可视化的网页制作工具，目前都包含了站点远程维护功能。其中，对于要使用 FrontPage 管理的网站，不仅要在服务器端添加 FrontPage 扩展，还必须在 IIS 中对网站属性进行设置，可见 FrontPage 只能远程维护 IIS 中的网站，详细介绍请参见第 4 章的内容。

## 思 考 题

1. 什么是 Web 服务器？有哪些主流的 Web 服务器产品？
2. 什么是 IIS？IIS 6.0 包括哪些可选组件？简述它们的功能。
3. 简述在 IIS 中 Web 站点的创建过程。
4. 在连接新建的 Web 站点时出现下面的“输入网络密码”对话框，为什么？如何解决？




5. 在 Windows 平台下，如何实现 Web 服务器的远程管理？
6. 什么是虚拟目录？使用虚拟目录有何好处？
7. 当连接到一个 IIS 中的 Web 站点，浏览一个 ASP 页时，显示“网页无法显示”提示页面，并且在页面中提示：“您试图从目录中执行 CGI、ISAPI 或其他可执行程序，但该目录不允许执行程序”，为什么？如何解决？

8. 为什么说 Apache 服务器是使用最广泛的 Web 服务器？它有哪些主要特点？

9. Java 运行环境包括哪些内容？安装 JDK 后，需要设置哪些系统环境变量？简述设置每个环境变量的目的。

10. 安装一次 Tomcat 6，简述 Tomcat 目录结构中各个目录的功能。

11. 在 Tomcat 6 中，简述主配置文件 conf/server.xml 的基本功能。

12. 在 Web 应用中，简述 WEB-INF 文件夹的作用，Web 应用配置文件 web.xml 的功能是什么？

13. Tomcat 必须和 Apache 集成才能使用吗？为什么？

14. Tomcat 对 JSP 页面是如何处理的？

15. 在一台 Windows 服务器上，能同时安装 IIS 和 Tomcat 吗？如果能，在 IIS 下创建的网站和 Tomcat 下的网站能同时运行吗？如何配置？

16. 要实现对 Web 站点的远程管理，有哪些方式？

第3章

# HTML和XML基础

标记语言是 Web 应用的基础，标记语言是由内容和标记组成的。在 Web 应用中，所有的 Web 页面都是以标记语言书写的具有特定格式的文档。因此，无论是 Web 应用还是开发，都应该对标记语言有一个基本的认识。

标记语言主要有 HTML 和 XML 两种，虽然都称为标记语言，但两者有着本质的区别。本章将对两者进行深入的讲解，特别是对 XML 的本质、XML 相关技术规范的功能、相互之间的关系进行深入的讲解，从而使得大家对 XML 技术有一个正确的认识。

## 3.1 标记语言及其发展

广义上的标记语言可以理解为对内容进行描述的规范或标准，例如在出版印刷行业，编辑人员在进行文档内容编辑时对内容所做的标记，通过这些标记符号来表达对内容的修改信息。在 Web 中，超文本标记语言 HTML 则通过标记来定义内容在浏览器中的显示样式。随着语义 Web 的发展以及异构环境下的数据交换需求，扩展标记语言 XML 则主要用于对数据结构和数据内容进行标记，成为重要的数据表达、交换和集成标准，对数据的显示则通过其他相关语言来完成，实现了数据内容和显示的分离。

### 3.1.1 标准通用标记语言（SGML）

标准通用标记语言（Standard Generalized Markup Language，SGML）是一个用来定义在电子表格中如何对文件的结构和内容进行描述的国际标准（ISO-8879）。时间可以追溯到 1969 年，当时美国 IBM 公司的研究人员开始设计一种名为 GML（Generalized Markup Language）的语言，在印刷、统计等需要大规模数据处理的行业和部门的支持下，这项研究工作持续了十几年，于 1980 年推出了 SGML 语言，并于 1986 年获得国际标准化组织 ISO 的批准。其后，SGML 的发展较为平稳，并不为其领域之外的人们所广泛了解。直至 1991 年，当超文本标记语言 HTML 问世之后，人们才开始认识 SGML。

为了满足各种不同的页面表达需要，SGML 设计的非常复杂，SGML 的正式规范达 500 多页。因此使用起来很不方便，使得它未能得到普及和大规模的应用。虽然 SGML 没有 HTML 那样应用广泛，但是 SGML 定义了标记语言的基本概念，奠定了标记语言的技术

基础。

现在,在 Web 中普遍应用的 HTML 和 XML 都是在 SGML 的基础上开发成功的,可以说它们都是 SGML 的一个子集。作为互联网信息共享的技术规范,标记语言对互联网的发展起到了巨大的推动作用。

### 3.1.2 超文本标记语言(HTML)

超文本标记语言(HTML)起源于标准通用标记语言(SGML),由世界上最大的粒子物理研究实验室欧洲核子研究中心 CERN(the European Organization for Nuclear Research)于 1991 年首先提出,是推动 Web 迅速发展的原动力。

在互联网发展的早期,为了在各种网络环境之间、不同文件格式之间进行交流,在 SGML 基础上,CERN 提出了超文本标记语言(Hyper Text Markup Language,HTML)的概念。HTML 是一种用来制作超文本文档的简单标记语言,它定义了一组标记符号(tag),对文件的内容进行标注,指出内容的输出格式,如字体大小、颜色、背景颜色、表格形式、各部分之间逻辑上的组织等,从而实现了文件格式的标准化。简单地说,HTML 文件包含了文档数据和显示样式两部分,其中文档数据是显示在 Web 浏览器中的数据内容,显示样式则规定了这些内容在浏览器中以何种格式、样子呈现给用户。通过统一使用支持 HTML 的浏览软件,用户可以在任意异构的网络环境中阅读同一个文件,得到相同的显示结果,并可以对文件进行跳跃式阅读,展现了很强的表现力。

HTML 主要版本和发布时间如下:

(1) HTML 2.0,Internet 工程任务组中的 HTML 工作组开发完成了 HTML 2.0,于 1996 年发布。

(2) HTML 3.2,W3C 于 1997 年 1 月 14 日将其列为推荐版本,在 HTML 2.0 标准中添加了诸如字体、表格、Java 程序、浮动、上标、下标等特征。

(3) HTML 4.0,W3C 于 1997 年 12 月 18 日将其列为推荐版本,第二个稍作修正的 HTML 4.0 版本于 1998 年 12 月 24 日发布。HTML 4.0 中最重要的特征是引入了样式表 CSS 技术。

(4) HTML 4.01,W3C 于 1999 年 12 月 24 日将其列为推荐版本,是 HTML 4.0 的升级版本,它对原版本做出了部分修正。

### 3.1.3 可扩展 HTML 规范 XHTML

在 HTML 的发展过程中,暴露出一些影响其发展的缺陷,例如:HTML 的标记固定,HTML 只是一种表现技术,不能表达语义;不能适应现在越来越多的网络设备和应用的需要,比如手机、PDA、信息家电都不能直接显示 HTML;由于 HTML 代码不规范、臃肿,浏览器需要足够智能和庞大才能够正确显示 HTML;数据与表现混杂,页面要改变显示,就必须重新制作 HTML。为此,W3C 不再继续开发 HTML。

2000 年底,W3C 制定了可扩展 HTML,即 XHTML,它是 HTML 向 XML 过渡的一个桥梁。2000 年 1 月 20 日发布 XHTML 1.0,XHTML 1.0 是 HTML 4.01 基于 XML 的形

式。2002 年 8 月 5 日，发布 XHTML 2.0 的第一个工作草案，其最大的特点就是取消了向后兼容性，去除了原先版本中的一些标记。例如，不再支持使用很少的<em>(强调)和<strong>标记等，这就使得原先的一些网页在 XHTML 2.0 规范的浏览器中不能正确显示。

需要说明的是，在 XHTML 的研发过程中，不确定的东西还很多，作为 HTML 向 XML 的一种过渡技术，唯一确定的就是它要更好地表达文档中的语义和结构，将文档的内容和表现技术更好地分离，更好地实现互联网中的数据交换和展示。

### 3.1.4　可扩展标记语言(XML)

HTML 的不足推动了 XML(eXtensible Markup Language)的产生和发展，其核心思想是实现数据和显示的分离。1998 年 2 月 10 日，XML 工作组正式向 W3C 提交了 XML 的最终推荐标准，这就是 XML 1.0 标准。XML 规范定义了标记语言的主要特征，例如 DTD、XML 架构等基本要素，这些要素可以很好地用于定义数据，实现异构环境下的数据交换。

对 XML 文档内容的显示、查询及操作则通过其他一系列的规范来实现，这些相关的规范包括：可扩展样式语言 XSL、XML 路径语言 XPath、XML 查询语言 Xquery、可扩展连接语言 XLL 以及 XML 文档对象模型 DOM 与简单应用程序接口 SAX 等，通过这些规范来实现对 XML 文档的显示及其他各种操作。

## 3.2　超文本标记语言 HTML

超文本标记语言 HTML 是在 SGML 基础上发展起来的，是互联网中应用最为广泛的标记语言，被称为 World Wide Web 的通用出版语言。在互联网中，绝大多数的网页都是通过 HTML 标记语言书写的，所有的 Web 浏览器都支持 HTML 规范，并能很好地显示 HTML 网页文件。即使是 XML 文档，也通常需要通过 XSLT 转化为 HTML 格式进行显示。

### 3.2.1　HTML 标记语法和文档结构

HTML 文档是纯文本文件，由“显示内容”和“控制语句”两部分组成。控制语句描述了显示内容以何种形式在浏览器中显示，并负责客户与服务器之间的信息交换。控制语句以“标记(Tag)”形式出现，以区分于显示内容。

标记被封装在小于号(<)和大于号(>)构成的一对尖括号之中，标记一般分为首标记和尾标记，它们成对出现。首标记用于开启某种形式的显示，尾标记用于关闭首标记开启的功能。例如：<u>text with underline</u>，首标记<u>开启下划线功能，尾标记</u>关闭下划线功能。该语句在浏览器中将把文本串“text with underline”加上下划线显示。有的控制语句仅需一个标记，没有尾标记，如：<br>标记表示换行。

### 1. 标记类型与标记属性

标记分为“单标记”和“双标记”两种类型。“单标记”是指只需单独使用就能完整地表达意思的一类标记,这类标记的语法是:

<标记>

常用的单标记有换行标记<br>、水平线标记<hr>等。

另一类标记称为“双标记”,由“首标记”和“尾标记”两部分构成,必须成对使用。首标记告诉 Web 浏览器从此处开始执行该标记所表示的功能,而尾标记告诉 Web 浏览器在这里结束该功能。首标记前加一个斜杠(/)即成为尾标记。这类标记的语法是:

<标记>文档内容</标记>

其中“文档内容”部分就是要被这对标记施加作用的部分。例如,如果需要标记一个文本超链接,则将文本放在双标记<a>…</a>中即可:

```
<a>link text</a>
```

许多单标记和双标记的首标记内可以包含一些属性,其语法是:

<标记 属性 1 = "属性值" 属性 2 = "属性值" 属性 3 = "属性值"…>

各属性之间无先后次序,属性之间用空格分开。属性也可省略(即取默认值),属性值两侧的双引号(")可以省略不写。例如:单标记<hr>表示在文档当前位置画一条水平线(horizonta line),一般是从窗口中当前行的最左端一直画到最右端。另外,<hr>标记还有 size、align、width 等属性,例如:可写作:<hr size="3" align="center"width="50%">。其中 size 属性定义线的粗细,默认为 1;align 属性表示对齐方式,可取 left(左对齐,默认值)、center(居中)和 right(右对齐);width 属性定义线的长度,可取相对值(整个窗口的百分比),也可取绝对值(屏幕像素点的个数,如 width=“400”),默认值是“100%”。

### 2. 文档结构

一个 HTML 文档以<html>标记开始,以</html>标记结束,表示这对标记间的内容是 HTML 文档。HTML 文档的中间分成文件头和文件体两个部分,由相应的标记来区分。HTML 文档总体结构如下:

```
<html>
<head>
     头部信息
</head>
<body>
     文档主体
     (语句部分)
</body>
</html>
```

其中,<head>…</head>之间是文件头,由一系列子标记构成,如定义文档标题的<title>标记等,若不需头部信息则可省略此标记。<body>…</body>之间是文件体,

表示正文内容的开始，<body>标记一般不能省略。

### 3.2.2　文件头标记及子标记

在 HTML 文档中，<head>…</head>标记对之间的部分称为文件头。文件头的内容通常是一些辅助性的标记，为浏览器、搜索引擎等提供信息，例如：设置网页内容字符编码、网页关键字等，便于浏览器对网页的正确显示和搜索引擎记录网页关键字。

文件头中主要的标记对有<title></title>、<script></script>等，下面分别介绍。

#### 1. <title> </title> 标记

出现在<head>…</head>标记对内，用于标识网页主题，其中的内容将在浏览器的标题栏中显示，不出现在页面内。

例如：

```
<title>欢迎使用 GSL 3.0 系统</title>
```

#### 2. <meta> 标记

它是 HTML 文档文件头<head>…</head>标记内的一个辅助性标记，往往不引起用户的注意，但是它对于网页是否能够被搜索引擎检索、提高网页在搜索列表的排序起着关键的作用，是一个非常有价值的标记。

<meta>标记为单标记，没有尾标记。<meta>标记共有两个属性，分别是 http-equiv 属性和 name 属性，不同的属性又有不同的参数值，这些不同的参数值实现了不同的网页功能。

1）name 属性

name 属性主要用于描述网页，与之对应的属性值为 content，content 中的内容主要是便于搜索引擎查找信息和分类信息用的。meta 标记的 name 属性语法格式是：

```
<meta name = "参数" content = "具体的参数值">
```

其中 name 属性主要有以下几种参数值：

• keywords(关键字)：keywords 用来告诉搜索引擎该网页的关键字是什么。

例如：<meta name="keywords" content="E-learning,ontology">

• description(网站内容描述)：description 用来告诉搜索引擎网站的主要内容。

例如：<meta name="description" content="This page is about E-learning etc.">

• author(作者)：标注网页的作者。

例如：<meta name="author" content="brion@mail.abc.com">

• robots(机器人向导)：robots 用来告诉搜索机器人需要索引的页面有哪些。content 的参数有 all、none、index、noindex、follow、nofollow，默认值为 all。

例如：<meta name="robots" content="none">

2）http-equiv 属性

http-equiv 相当于 HTTP 的文件头，向浏览器传回一些有用的信息，以帮助正确显示网

页内容，与之对应的属性值为 content，content 中的内容其实就是各个参数的变量值。meta 标记的 http-equiv 属性语法格式是：

```
<meta http-equiv="参数" content="参数变量值">
```

其中 http-equiv 属性主要有以下几种参数：

- content-type(显示字符集的设定)：设定页面使用的字符集。

例如：<meta http-equiv="content-type" content="text/html; charset=gb2312">

- expires(期限)：用于设定网页的到期时间。一旦网页过期，必须到服务器上重新下载。

例如：<meta http-equiv="expires" content="Thur,8 May 2008 18:18:18 GMT">

- pragma(Cache 模式)：禁止浏览器从本地计算机的缓存中访问页面内容。

例如：<meta http-equiv="pragma" content="no-cache">，该种设定访问者将无法使用脱机浏览功能。

- refresh(刷新)：自动刷新并指向新页面。

例如：<meta http-equiv="refresh" content="60; url=new.htm">

则浏览器将在 60 秒后自动转到 new.htm。利用该功能，可以显示一个封面提示页面，在若干时间后，再自动转移到其他页面。

如果不设置 URL 项，浏览器则刷新本页，这就实现了 Web 聊天室定期刷新的特性。

- window-target(显示窗口的设定)：强制页面在当前窗口以独立页面显示。

例如：<meta http-equiv="window-target" content="_top">，可以用来防止别人在框架里调用自己的页面。

### 3. <base> 标记

<base>标记定义了文档的基础 URL 地址，在文档中所有的相对地址形式的 URL 都是相对于这里定义的 URL 而言的。一篇文档中的<base>标记不能多于一个，必须放于头部，并且应该在任何包含 URL 地址的语句之前。

<base>标记包含如下属性：

1) href 属性

href 属性指定了文档的基础 URL 地址，该属性在<base>标记中是必须存在的。

例如：如果希望将文档的基础 URL 定义为“http://www.abc.com”，则可以使用如下语句：

```
<base href="http://www.abc.com">
```

当定义了基础 URL 地址之后，文档中所有引用的 URL 地址都从该基础 URL 地址开始，例如，对于上面的语句，如果文档中一个超级链接指向 gsl/welcome.htm，则它实际上指向的是如下 URL 地址：http://www.abc.com/gsl/welcome.htm。

2) target 属性

target 属性同框架一起使用，它定义了当文档中的链接被单击后，在哪一个框架中展开页面。如果文档中超级链接没有明确指定展开页面的目标框架集，则就使用这里定义的地址代替。常用的 target 的属性值有：

- _blank：表明在新窗口中打开链接指向的页面。
- _self：在当前文档的框架中打开页面。
- _parent：在当前文档的父窗口中打开页面。
- _top：在链接所在的完整窗口中展开页面。

例如：<base target="_blank"> 表明页面上所有的链接都在新窗口打开。

#### 4. <link> 标记

<link>标记定义了文档之间的包含。在 HTML 的头部可以包含任意数量的<link>标记。有些浏览器并不能很好地处理 link 属性，因此不建议使用它。

<link>标记带有很多属性，下面是一些常用的属性：

(1) type：用于指定被包含的文件类型。例如，text/css 是指包含一个层叠样式表文件。

(2) href：指向被包含资源的 URL 地址。

(3) title：一个字符串，用于描述该链接关系。

(4) rel：定义 HTML 文档和所要包含资源之间的链接关系，可能的取值很多，最为常用的取值是 stylesheet，用于包含一个固定首选样式表单。

例如，如果文档包含一个外部的 CSS 文件，在文档头部应该定义如下：

```
<link type = "text/css" rel = "stylesheet" href = "mystyle.css">
```

#### 5. 背景音乐标记<bgsound>

在文档头内，还可以定义背景音乐，标记为<bgsound>，用以插入背景音乐，可插入的音频文件类型有：Wave 文件(*.wav)、Midi 序列文件(*.midi)、Real Audio 文件(*.ram、*.ra)、AIFF 声音文件(*.aif、*.aifc、*.aiff)、AU 声音文件(*.au、*.snd)。

<bgsound>标记为单标记，一般形式是：

```
<bgsound src = " " autostart = "" loop = "">
```

标记的属性有：

(1) src 属性：给出音乐文件的 URL 值。

(2) autostart 属性：设置音乐文件播送结束后的处理，如果为 true，则自动播放音乐，为 false 则结束播放，默认值为 false。

(3) loop 属性：设置是否自动反复播放，loop=2 表示重复两次，Infinite 表示重复多次。

### 3.2.3　文件体标记及其属性

在<body>…</body>标记对之间的部分称为 HTML 文档的文件体。文件体中描述的是浏览器中显示的内容。文件体由一系列的控制语句构成，在<body>…</body>标记对之间可包含<p>…</p>、<img>、<a>…</a>等标记，它们所定义的文本、图像以及超链接等将会在浏览器中显示。

<body>标记是一个非常重要的标记,含有大量的属性,许多重要的网页功能都是通过<body>标记的属性实现的,标记属性可分为一般属性和事件属性两大类。

### 1. 一般属性

<body>标记的一般属性用于页面的一般性设置(见表 3-1)。

**表 3-1 <body>标记一般属性**

| 属　性 | 用　途 | 举　例 |
|---|---|---|
| bgcolor="#rrggbb" | 设置背景颜色 | <body bgcolor="red">红色背景 |
| text="#rrggbb" | 设置文本颜色 | <body text="#ff0000">红色文本 |
| link="#rrggbb" | 设置未阅读过的超文本链接颜色,默认值是蓝色 | <body link="blue">链接为蓝色 |
| vlink="#rrggbb" | 设置阅读过的超文本链接颜色,默认值是紫色 | <body vlink="#ff0000">红色 |
| alink="#rrggbb" | 设置动作中的超文本链接颜色,默认值是紫色 | <body alink="yellow">设为黄色 |
| leftmargin、topmargin | 设置页边距 | <body leftmargin="0" topmargin="0"> |
| background | 设置页面背景图片的 URL | background="image/10.jpg" |
| bgproperties | 设置成 fixed,则背景图案不滚动 | bgproperties=fixed |

颜色的设置可以通过 HTML 语言所给定的颜色常量名,或者 RGB(红、绿、蓝三色的组合)颜色设置,例如"#ff0000"表示红色。各个属性可以结合使用,如:<body bgcolor="red" text="#0000ff">,设置网页的背景色为红色(red),文本为蓝色("#0000ff")。

### 2. 事件属性

当一个 Web 文档被加载显示或者退出(关闭),当进行移动窗口或改变文档窗口大小等操作时,会发生相应的事件,这些事件在<body>标记中通过事件属性来表达。<body>标记常见的事件属性见表 3-2。

**表 3-2 <body>标记中的事件属性**

| 事　件 | 触发条件 | 事　件 | 触发条件 |
|---|---|---|---|
| onLoad | 页面下载完成时触发 | onMouseMove | 鼠标移动时触发 |
| onUnload | 退出页面时触发 | onDblClick | 鼠标双击时触发 |
| onFocus | 页面窗口获得焦点时触发 | onMouseDown | 鼠标被按下时触发 |
| onBlur | 页面窗口失去焦点时触发 | onKeyDown | 键被按下时触发,按键的 ASCII 码值保存在 window.event.keyCode 中 |
| onResize | 窗口改变大小时触发 | onKeyPress | 键被按下然后被释放时触发 |
| onScroll | 单击滚动条时触发 | onKeyUp | 键被释放时触发 |

在上述事件中,有些事件是<body>标记特有的,有些事件可能存在于多个不同的标记中。

事件属性的值往往是一个 JavaScript 函数，来完成 Web 编程任务。在 FrontPage、Dreamweaver 等工具软件中，可以通过行为面板（在 FrontPage 2003 中，对应"格式"、"行为…"菜单命令）显示一个标记支持的行为事件，并且可自动生成简单的行为 JavaScript 代码，从而减少用户书写程序的工作量。具体应用参考后面的章节。

**【例 3-1】** 一个简单的 HTML 文档示例。

在 Windows 中，用"记事本"程序编写一个简单的 HTML 文件，如图 3-1 所示。




图 3-1　用"记事本"程序创建 HTML 文档

双击上述的 HTML 文档，文档将在浏览器中打开，显示结果如图 3-2 所示。

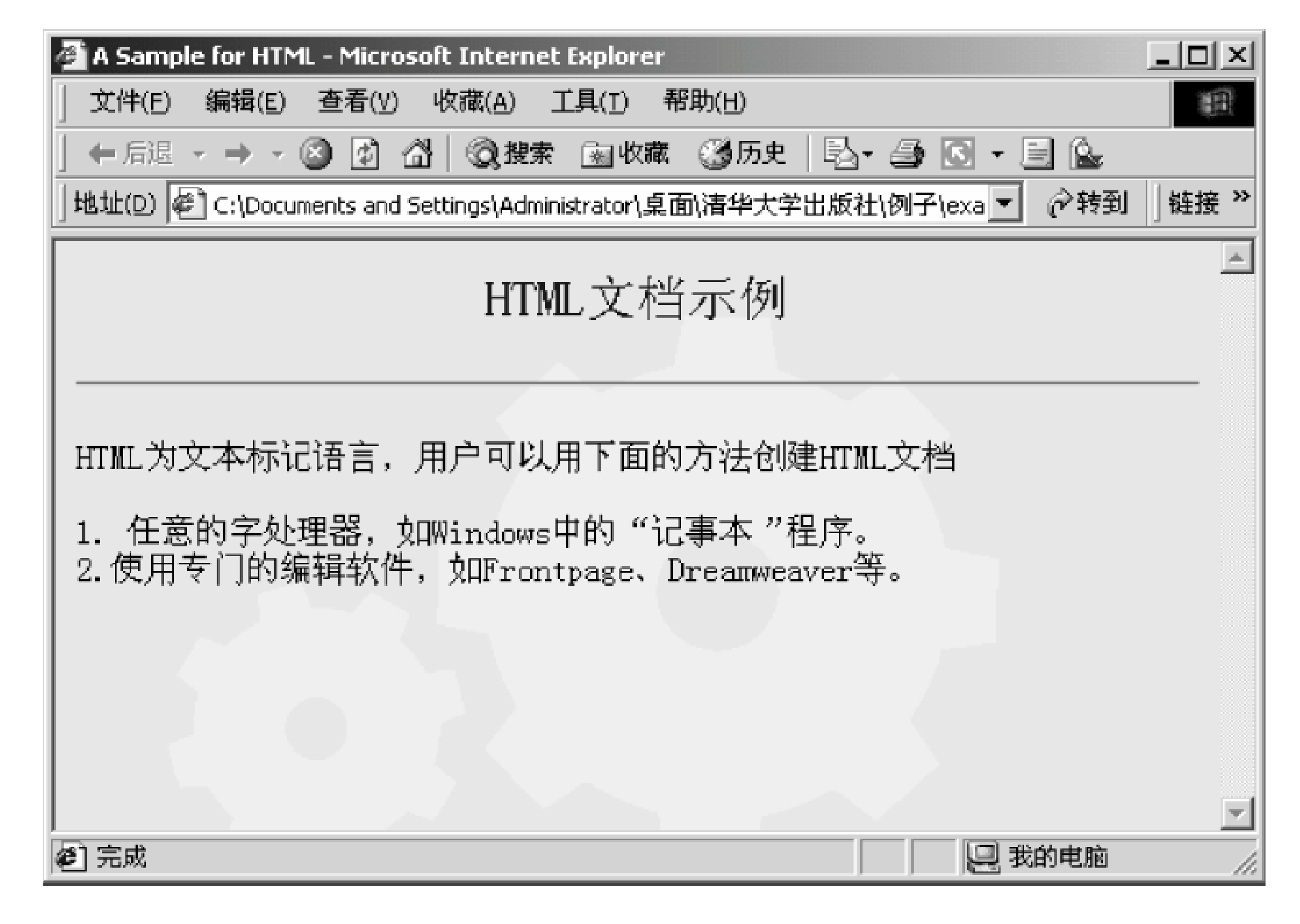



图 3-2　HTML 文档在浏览器中的显示界面

### 3.2.4 文档内容常用标记

HTML 文档体是由一系列的标记语句组成的，这些标记标记了文档内容的显示形式，可以说它们是<body></body>标记的子标记。对于大多数标记，都可以对标记设置相应的一般属性和事件属性，来改变标记默认的显示样式或激活事件行为函数。由于本书篇幅所限，对标记的属性将不做详细介绍。

#### 1. 标题、段落标记

在<body>…</body>标记对之间的内容将在浏览器中显示，对显示内容的格式指定是通过各种格式标记来定义的。常用的格式标记有：

(1) 标题标记<h1></h1>…<h6></h6>：一级到六级标题标记，对应的字体逐渐缩小。

(2) 段落标记<p>…</p>：标记一个段落，输出位置转到下一行开始，并增加一个空行。

(3) 回车换行标记<br>：将输出位置转到下一行的开始。

(4) 水平线标记<hr>：插入一条水平线。

其中每一个标记都可以设置相应的属性，除了一般属性外，还可以设置事件属性。例如，对于段落标记<p>，除了具有 align 等一般属性外，还具有 onclick、ondbclick、onmouseover、onmousemove、onkeydown、onkeypress、onkeyup 等事件属性，详细介绍略。

#### 2. 文本格式标记

用来定义文本输出的字体和格式，如斜体字、黑体字、下划线等。常用的文本格式标记有：

(1) 字体标记<font>…</font>：常用的属性有 face、size 和 color，分别设置字体、文字大小和颜色。

(2) <b>…</b>、<i>…</i>、<u>…</u>标记：分别对应为粗体、斜体和下划线标记。

上述标记可以联合使用，例如：

```
<b><font face="宋体" size="3" color="#0000FF"><i><u>字体标记一</u></i>
</font></b>
```

显示效果为：

***字体标记一***

(3) <big>…</big>、<small>…</small>、<sub>…</sub>、<sup>…</sup>标记：分别为字体放大、缩小、上标和下标显示标记。

(4) <strong></strong>、<em></em>、<tt></tt>、<cite></cite>标记：<strong>…</strong>标记和<b>…</b>标记类似，都是字体加粗标记。<em>…</em>标记和<i>…</i>标记类似，是斜体强调标记。<tt>…</tt>为打字字体

Courier 字体，字母等宽标记。<cite>…</cite>为传记引述斜体效果标记。

### 3. 图像标记<img>

图像标记<img>为单标记，用以插入图像及设定图像属性，丰富多彩的图像可以提高网页的吸引力。<img>标记的属性很多，下面是一些常用的属性。

- align：设置图像的对齐方式，取值分别是“left”、“right”、“center”、“top”、“bottom”等。在图文混排时，这个参数很有用。
- id：指定的图片 id 号，用于对图像的程序访问。
- class：指定图像所属的类型。
- name：用于设定图像的名称，用于对图像的程序访问。
- src：设置插入图像的 URL 地址，即插入图像的路径和文件名。
- title：设置图像标题。
- alt：设置图像替代文字，主要用于在浏览器还没有装入图像（或关闭图像显示）的时候，此图像的显示信息。
- border：设置图片边框。
- height 和 width：分别用于设置图像的高度和宽度，可以与图片原来的宽度和高度不同。
- hspace 和 vspace：分别用于设置图像的左右边框大小和上下边框大小。
- ismap 和 usemap：在应用图像地图（map）时使用。ismap 表示图像地图的数据存放在服务器中，当鼠标在图像上的某个区域上时，可以将此区域的坐标传送给服务器处理。usemap 则用于设定图像地图的名称。

除了上述的一般属性外，<img>标记还有大量的鼠标和键盘事件属性，例如：onload、onclick、ondbclick、onmouseover、onkeydown、onkeypress 等，在 FrontPage 等工具中，单击插入的图片，在行为面板中，可以查看图片所有的事件属性。

需要说明的是，<img>标记并不是真正地把图像加入到 HTML 文档中，而是将标记的 src 属性赋值为图形文件所在的路径及文件名（图像的格式可以为 gif 或 jpg）。

**【例 3-2】** 图像标记的应用示例。

下面代码显示一幅泰山图片，当鼠标移到图片上的时候，显示一个简单的介绍，鼠标移走后重新显示图片。

HTML 代码如下：

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>Taishan Introduction</title>
</head>
<body>
<img src="image/t01.jpg" width="200" height="150" border="1" align="left" name="tai"
     alt="泰山图片" vspace="10" hspace="10"
     onmouseover="tai.src='image/t02.gif'" onmouseout="tai.src='image/t01.jpg'" >
</body>
</html>
```

使用<img>标记还可以启动媒体播放机播放影视片断，代码如下：

```
<img dynsrc = "move/taishanfg.avi" start = "mouseover" width = "300" height = "200" controls
loop = "1" alt = "泰山风光介绍影像片断">
```

上面这段代码的效果是，当鼠标移开影像片断上时即开始启动媒体播放器播放影像一次，并且还设定了显示播放器控制面板，以便对播放过程进行控制。

在网页中加入影像片断与加普通图片基本相同，如它们都要指定来源、设定大小和替代文字等。但需要说明的是，加入影像片断时要使用 dynsrc 属性，且不能再在同一个<img>标记中使用“src”属性，否则将不能播放影像片断，只能显示替代文字。

#### 4. 超链接标记<a>…</a>

超链接标记<a>…</a>可以定义文本超链接和图像超链接。

1）文本超链接

定义文本超链接的一般格式为：

```
<a href = "url#name">字符串</a>
```

在文本超链接定义中，href 属性的值为被链接文档的 URL 地址，字符串为超链接显示的文字，一般显示为蓝色。当鼠标指向这个字符串时，鼠标变成手形，表示当前位置是一个超链接。name 是文档内的书签名，如果指定书签，则进入超链接文档后，将定位到书签所在的文档位置。如果 href="" 为空字符串，则定义一个空超链接，即不指向任何超链接位置。如果 href="#name"形式，则定位到当前文档的 name 书签位置。

2）图像超链接

图像超链接就是将一个图片定义为一个超链接，一般形式是：

```
<a href = "url#name "> <img src = "imgeurl"> </a>
```

当鼠标移到图片上方时，鼠标变成手形，表示该图片对应一个超链接。

3）定义书签

对于一个完整的文件，我们可以用它的 URL 来唯一地标识它，但对于一个文件的不同部分，我们怎样来标识呢？这就是书签的概念，它是通过链接标记来定义的。在文档内部可以定义书签，书签是通过<a>标记的 name 属性定义的。标识一个书签的方法为：

```
<a name = "bookmarkname">书签文本</a>
```

name 属性将放置该标记的地方标记为一个名为“bookmarkname”的书签，书签必须是一个全文唯一的标记串。有了书签后，href 属性除了指向一个网址外，还可以定位到网页内一个具体的书签，用法是 href="url#bookmarkname"，如果是同一个页内，可以写成 href="#bookmarkname"，同一网页内书签名不能重名。书签文本可以为空。

4）其他属性

除了 href 和 name 属性外，<a>标记有其他一些属性，这些属性包括：

- target 属性：定义超链接打开的目标窗口。一般形式是 target="window-name"，还可以取下面常量：_self(相同框架)，_blank(新建窗口)，_top(整页)，_parent(父窗口)。
- title 属性：属性值为一字符串，鼠标指向超链接时，鼠标右下角显示标题文本。

通过 title 和 href=""空超链接属性结合，可以产生特定的效果。对尚未完成的超链接显示一个提示信息，当超链接页面完成后，再给 href 属性赋具体的 URL 值，例如：

```
<a href = "" title = "have not compledted now">学习论坛</a>
```

如果在提示信息中需要换行，可以使用“”或“
”来完成换行输出，例如：

```
title = "提示：&#13;来宾无此权限"
```

**【例 3-3】** 超链接标记使用示例。

下述代码演示了超文本链接、在 HTML 文档中插入书签、图像链接的应用。

```
<html>
<head>
<meta http-equiv = "Content-Language" content = "zh-cn">
<title>Shandong Travel</title>
</head>
<body>
<p align = "center">美丽的山东</p>
<p><a name = "济南">美丽的泉城济南</a></p>
⋮
<p><a name = "青岛">美丽的海滨城市青岛</a></p>
⋮
<a href = "http://www.cnta.com/" title = "中国旅游网" target = "_blank">主要旅游网</a>
</body>
</html>
```

### 5. 影像地图标记<map> </map>

所谓影像地图，就是在一个图片上定义一系列区域，每个区域对应一个超链接。设置影像地图，首先要通过<img>标记插入一幅图片，并设置<img>的 ismap 属性，然后再定义相应的热点区域。使用影像地图的一般形式是：

```
<img src = "" usemap = "#mapname" ismap width = "" height = "">
<map name = "mapname">
<area href = "1.htm" shape = "circle" coords = "379,1212,79" target = "_blank" title = " ">
<area href = "2.htm" shape = "rect" coords = "224,159,274,180" target = "_blank" title = "" >
⋮
</map>
```

其中，<map>…</map>为影像地图标记，具有一个 name 属性，指定图像地图的名称，中间包含一系列 area 标记。<area>标记是影像地图标记<map>的子标记，为单标记形式。通过该标记可以在图像地图中设定作用区域(又称“热点”)，当用户鼠标移到某作用区域点击时，执行对应的事件处理函数。<area>标记的常用属性有：

- href 属性：用于设定该热点所链接的 URL 地址。
- alt 属性：用于设定热点的替代性文字。
- shape 和 coods 属性：shape 和 coords 是两个主要的参数，用于设定热点的形状和大小。

其基本用法如下：

shape="rect" coords="$x_1$,$y_1$,$x_2$,$y_2$"：表示设定热点的形状为矩形，左上角顶点坐标

为($x_1$,$y_1$),右下角顶点坐标为($x_2$,$y_2$)。

shape="circle" coords="$x_1$,$y_1$,r":表示设定热点的形状为圆形,圆心坐标为($x_1$,$y_1$),半径为 r。

shape="poligon" coords="$x_1$,$y_1$,$x_2$,$y_2$…":表示设定热点的形状为多边形,各顶点坐标依次为($x_1$,$y_1$)、($x_2$,$y_2$)、($x_3$,$y_3$)……。

<area>标记是在图像地图中划分作用区域的,其划分的作用区域必须在图像地图的区域内,所以在用 <area> 标记划分区域前必须用<map>标记来设定图像地图的作用区域,并为指定的图像地图设定名称。

设置图像地图通常是利用 FrontPage 等工具实现的,例如,在 FrontPage 2003 中,首先插入一幅图片,然后在图片上右击鼠标,执行"显示图片工具栏"命令,即可定义热点区域。

对于 Web 编程,经常在图像地图上增加 JavaScript 代码来实现一些用户逻辑。实际上 JavaScript 几乎把 area 标记与超文本链接标记<a>…</a>一样看待。所以我们也可以把 onClick、onMouseOver 和 onMouseOut 等这些事件加到 area 标记中。

#### 6. 走马灯标记<marquee>…</marquee>

"走马灯"标记<marquee>…</marquee>用于标记一行或多行滚动的文本,也可以将文本带有超链接,以增加网页的动态效果。现在主流的浏览器,如 IE、Maxthon、Firefox 等均支持<marquee>标记。

<marquee>标记常用的属性有:

- align 属性:设定活动字幕的位置,取值可以是 left、center、right、top 或 bottom。
- bgcolor 属性:设定活动字幕的背景颜色,一般是十六进制数。
- direction 属性:设定活动字幕的滚动方向,取值可以是 left、right、up 或 down。
- behavior 属性:设定滚动的方式,主要有三种方式:behavior="scroll"表示由一端滚动到另一端;behavior="slide"表示由一端快速滑动到另一端,且不再重复;behavior="alternate"表示在两端之间来回滚动。
- height 和 width 属性:设定滚动字幕的高度和宽度。
- hspace 和 vspace 属性:设定滚动字幕的左右边框和上下边框的宽度。
- scrollamount 属性:设定活动字幕的滚动距离。
- scrolldelay 属性:用于设定滚动两次之间的延迟时间。
- loop 属性:用于设定滚动的次数,当 loop=−1 表示一直滚动下去,直到页面更新。

默认情况下,<marquee>标记是向左滚动无限次,字幕高度是文本高度,水平滚动的宽度是当前位置的宽度;垂直滚动的高度是当前位置的高度。

由于<marquee>标记只能作用于一段(<p>…</p>)文本,因此活动字幕为多行时,分行时只能用<br>标记,不能用<p>标记。例如:

```
<marquee onmouseover = this.stop() onmouseout = this.start() scrollAmount = 1 scrollDelay = 60
direction = up width = 150 height = 200>
活动字幕内容第一行<br>
活动字幕内容第二行<br>
活动字幕内容第三行<br>
</marquee>
```

上述代码，将在一个区域内垂直地滚动多行，鼠标指向时停止滚动，离开时继续滚动。

另外，字幕中也可以加入图像，代码如下：

```
<marquee><img src = "image/logo.gif" width = "20" height = "20">欢迎光临</marquee>
```

如果希望滚动的内容带有超链接，可以将内容用<a>…</a>标记，即：<marquee><a href="0">活动字幕内容</a></marquee>。

注意，如果是使用 FrontPage 编辑 HTML 文档，对于多行和插入图片的<marquee>，FrontPage 会自动地处理成一行或者把<img>标记放到<marquee>…</marquee>标记的外面，此时需要通过记事本程序手工处理，将多行或图片放回到<marquee>…</marquee>标记内部，实现多行或者带有图片的滚动。

#### 7. 注释标记

注释标记一般的形式是<!--注释性文字-->，用于在 HTML 文档中书写说明性文字，注释文字可以多行，内容在浏览器中不显示。

### 3.2.5　表格

表格(table)是网页制作中安排布局最好的工具，因为表格不但可以很好地安排文本或图像的显示位置，而且还可以任意进行背景和前景颜色的设置。

#### 1. 表格标记<table>…</table>

<table>…</table>标记对用来创建一个表格，每个<table>…</table>标记对之间包含若干<tr>…</tr>，一个<tr>对应表格的一行。每一个<tr>…</tr>标记对又包括若干个<td>…</td>标记对，它对应一行中的一个单元格。

表格的默认属性设置为无边框，并根据内容自动设定表格大小。要修改表格的默认外观特性，需要设置相应的<table>标记属性，表格属性见表 3-3。

**表 3-3　<table>标记属性列表**

| 属　性 | 用　途 |
| --- | --- |
| bgcolor | 设置表格的背景色 |
| background | 设置背景图片 |
| border | 设置边框的宽度，若不设置此属性，则边框宽度默认值为 0，即无边框，无格线 |
| bordercolor | 设置边框的颜色 |
| bordercolorlight | 设置边框明亮部分的颜色(当 border 的值大于等于 1 时有效) |
| bordercolordark | 设置边框昏暗部分的颜色(当 border 的值大于等于 1 时有效) |
| cellspacing | 设置表格单元格之间空间的大小，默认值是 2 |
| cellpadding | 设置表格单元格边框与其内部内容之间的距离 |
| width | 设置表格的宽度，可以是像素值如 width="200"，或窗口总宽度的百分比，如 width="80%" |
| height | 设置表格中单元格的高度 |
| align | 设置表格的浮动对齐方式，只有 left 和 right 两种对齐方式 |

在表格属性表中，有关宽度、大小的单位用绝对像素值，而有关颜色的属性使用十六进制 RGB 颜色码或 HTML 语言给定的颜色常量名。

因为许多浏览器不支持<table>标记中的 align 属性值 center，因此往往要通过<div align="left|center|right">…</div>标记来设置整个表格的页面布局的居中对齐方式。如果<table>标记同时含有 align 属性，以<table>标记内的 align 属性设置显示。

### 2. 行、列和列标题标记<tr> </tr>、<td> </td>、<th> </th>

一个表格由若干行(Row)构成，每一行又由若干个单元格(Cell)组成，另外一个表格还可能具有一个标题(Head)。

根据上述表格的结构，在<table>…</table>标记对之间，用<tr>…</tr>标记对来创建表格中的每一行(Row)，表有多少行就有多少个<tr></tr>标记对；<td>则填充由<tr>和<th>组成的表格，<td></td>标记对用来创建表格中一行中的一个单元格，<td></td>标记对之间输入单元格中要显示的内容。

此外，<tr>还有 align 和 valign 属性，分别表示水平对齐和垂直对齐方式。<td>具有 width、colspan、rowspan 和 nowrap 属性。width 是单元格的宽度，单位用绝对像素值或总宽度的百分比；colspan 设置一个单元格跨占的列数(默认值为 1)，rowspan 设置一个单元格跨占的行数(默认值为 1)；nowrap 禁止单元格内的内容自动换行。

### 3. 表格标题标记<caption>…</caption>

标记表格标题，具有 align 和 valign 参数，设置水平和垂直对齐方式。

**【例 3-4】** 使用表格示例。

```
<html>
<head>
<title>表格标记综合示例</title>
</head>
<body>
    <table border = "1" cellpadding = "0" bgcolor = "#C0C0C0" width = "400" height = "75" >
        <caption>
         <p style = "margin - right: 16"><font size = "5" color = "#0000FF">学生成绩登记
表</font>
        </caption>
        <tr>
            <td align = "center" valign = "middle" width = "40%" height = "30">学   号</td>
            <td align = "center" valign = "middle" width = "20%" height = "22">姓   名</td>
            <td align = "center" valign = "middle" width = "20%" height = "22">高等数学</td>
            <td align = "center" valign = "middle" width = "20%" height = "22">英   语</td>
        </tr>
        <tr>
            <td align = "center" valign = "middle" width = "40%">2008000001</td>
            <td align = "center" valign = "middle" width = "20%">张三</td>
            <td align = "center" valign = "middle" width = "20%">95</td>
            <td align = "center" valign = "middle" width = "20%">90</td>
        </tr>
        <tr>
            <td align = "center" valign = "middle" width = "40%">2008000002</td>
```

```
            <td align="center" valign="middle" width="20%">李四</td>
            <td align="center" valign="middle" width="20%">90</td>
            <td align="center" valign="middle" width="20%">96</td>
        </tr>
              <tr>
            <td align="center" valign="middle" width="40%" rowspan="2">说   明</td>
            <td valign="middle" width="60%" colspan="3">成绩=平时*20%+期末*80%</td>
        </tr>
        <tr>
            <td valign="middle" width="60%" colspan="3">2006 年 1 月</td>
        </tr>
    </table>
</body>
</html>
```

在 Maxthon 浏览器中的显示结果如图 3-3 所示。

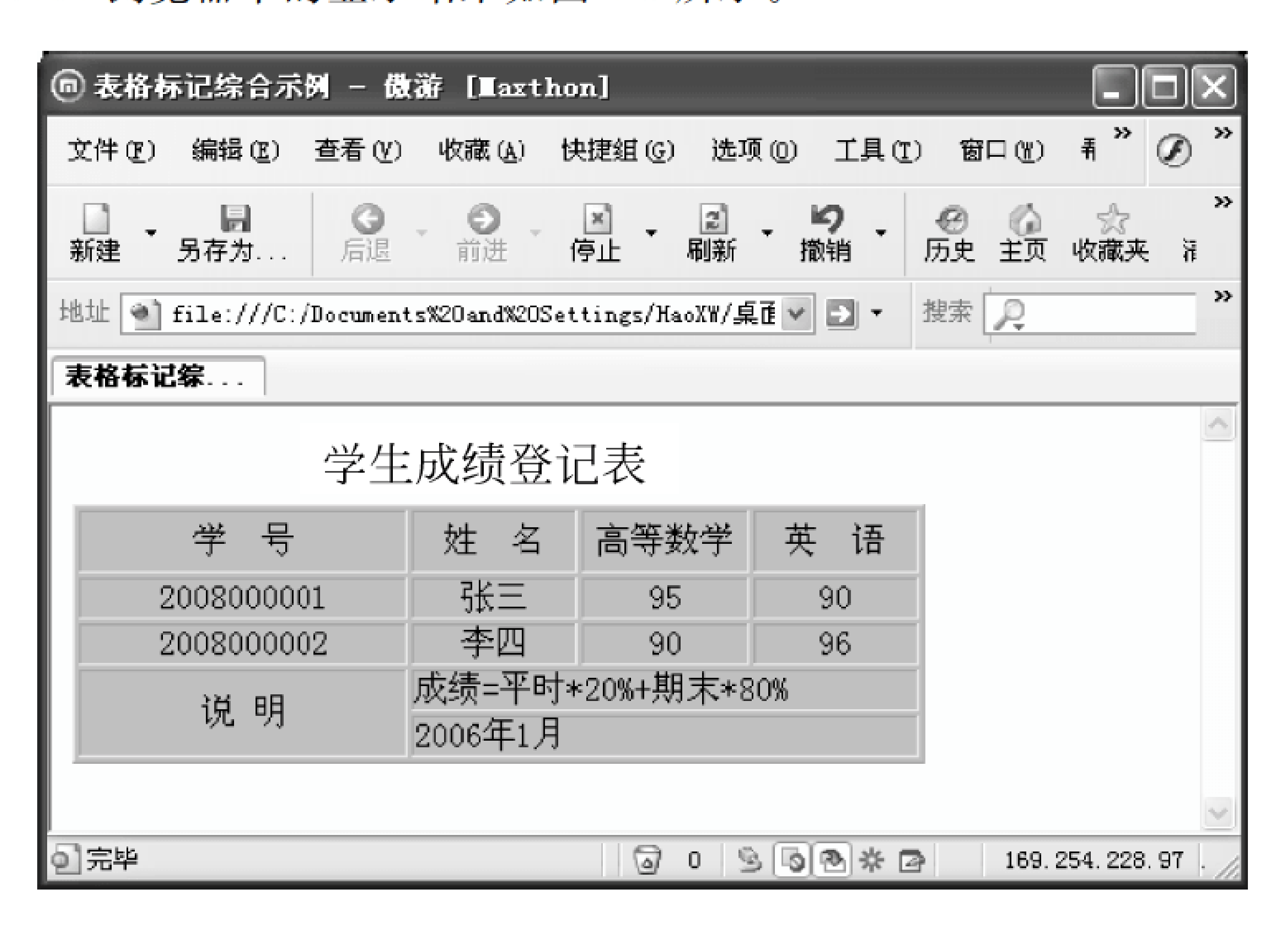



图 3-3　表格在 Maxthon 浏览器中的显示结果

在上述代码中，为了避免由于浏览器窗口大小变化引起单元格和表格大小变化，在<table>标记中，使用了 width 属性，指定绝对像素值，而不是相对比例。这样当窗口的大小变化时表格的大小不变。

在表格中还可以进行表格的嵌套定义，具体情况根据实际决定，同时为了提高浏览器的显示速度，一般不易定义一个大的表格，因为浏览器是等到整个表格下载完后才显示的。

关于表格更加复杂的操作，请参考第 5 章客户端开发。

## 3.2.6　表单

表单(form)在 Web 网页中用来给访问者填写信息，从而能获得用户信息，使网页具有交互能力。表单和 Windows 的对话框类似，是由若干控件组成的，用于实现和用户的交互。

当用户填写完信息后做提交(submit)操作,表单的内容将从客户端的浏览器传送到 Web 服务器,由 Web 服务器相应的服务器脚本程序处理。

对于表单及其包含的每一个元素,浏览器在进行解释时都会在内存中创建相应的对象,这和程序设计中的变量说明是一样的,这就为用户对表单输入数据的操作提供了技术上的保证。关于表单数据的处理将在本书第 5 章客户端开发和第 6 章服务端开发中进行讲解。本节只介绍 HTML 规范中表单及其元素的标记语法。

#### 1. 表单及输入类型

表单是 Web 中实现人机交互的主要界面,表单由控件构成,完成用户数据的输入。数据输入完成后,通过提交表单,表单数据被传送到 Web 服务器端,这些数据将被<form>标记的 action 属性所设置的程序处理。

1) 表单标记<form>…</form>

在 HTML 中,<form>…</form>标记对用来标记一个表单,即定义表单的开始和结束位置,在标记对之间的一切都属于表单的内容。<form>标记的常用属性有:

- name 属性:给出表单的名称,用于脚本编程,在一个网页中,可以包含多个表单。
- method 属性:method 属性用来定义服务器表单处理程序从表单中获得信息的方式,有 get 或 post 两种传输方式。

get 方法将数据打包放置在环境变量 QUERY_STRING 中作为 URL 整体的一部分传递给服务器。QUERY_STRING 变量可存储量是有限的,一般限制在 1KB 以下。

post 方法分离地传递数据给服务器表单处理程序,不需要设置 QUERY_STRING 环境变量,因此 POST 有更好的安全性,表单中数据的多少是任意的,因为这些数据从来也不分配到一个变量里。此外用 post 传递数据还有一个好处,它不会像 get 那样把传送的数据暴露在浏览器的地址栏中,如:myform.htm? Account=abc&pwd=123456。

- action 属性:设置表单处理程序的网络路径和程序名,当用户提交表单时,服务器将执行 action 属性所设置的程序。

在网页制作中,如果使用 FrontPage 设计表单,在生成的表单代码中,会看到 action 属性自动被赋值为"--WEBBOT-SELF--"。"WEBBOT-SELF"是 FrontPage 扩展所使用的伪代码。如果 Web 服务器上装有 FrontPage 扩展,在用户浏览网页时,服务器会自动把"WEBBOT-SELF"转换成适当的服务器上的路径。如果用户在浏览器上得到的就是"--WEBBOT-SELF--",说明 Web 服务器上没有安装 FrontPage 扩展。

2) 输入类型标记<input type="">

表单是由控件构成的,通过控件完成数据的输入。在 HTML 中,控件定义是通过输入类型标记完成的。输入类型标记的一般形式是:

```
<input type = "" name = "" value = "" …>
```

常用的属性有:

- type 属性:给出输入控件的类型,常用的控件类型有:text、textarea、radio、checkbox、button、image、hidden、password、file、submit/reset。
- name 属性:设置输入控件的名字,程序需要通过控件名称处理用户输入数据,同

时，设置控件名称可以增加 HTML 代码的可读性。

- value 属性：保存用户的输入和选择，服务器通过调用输入区域的 value 属性值来获得输入控件的数据。另外，用户可以通过 value 属性来指定输入区域的默认值。

根据输入类型 type 的不同，每种输入控件还有不同的其他属性。

### 2. 单行文本框输入标记

设置一个单行文本框输入，一般形式是：

```
<input type="text" name="" value=""…>
```

主要属性有：

- name 属性：文本框名称，便于程序获取用户输入。
- value 属性：存储文本框的取值，可以设一个初始值。
- size 属性：设置表示文本框的显示长度。
- maxlength 属性：maxlength 是文本框中输入的有效数据长度。

例如，html 代码如下：

```
<form name="myForm" method="POST" action="/custom/feedback.jsp">
用户账户：<input type="text" name="AccountStr" size="10" value="guest" maxlength="8">
</form>
```

显示效果为：

用户账户：guest

### 3. 密码文本框输入标记

输入密码文本框控件，和单行文本框控件相似，两者不同的是，使用输入密码文本框控件，当用户输入密码时，区域内将会显示“*”号。

一般形式是：

```
<input type=" password " …>
```

例如，HTML 代码如下：

```
<form name="myForm" method="POST" action="/progs/feedback.jsp">
密码：<input type="password" name="myPassword" size="10" maxlength="8">
</form>
```

显示效果为：

密码：*******

### 4. 多行文本框输入标记

多行文本框，又称滚动文本框。和其他的输入类型不同，它不是通过<input type=" " …>来指定的，它是一个双标记，一般形式是：

```
<textarea name=" " rows="" cols="">
input text
</textarea>
```

<textarea>标记的主要属性有：

- name 属性：多行文本框的名称，便于程序获取用户输入。
- rows 属性和 cols 属性：分别用来设置文本框的列数和行数，列与行以字符数为单位。

标记对之间的文本 input text 为显示的初始文本内容。

例如，HTML 代码如下：

```
<form name="myForm" method="POST" action="--WEBBOT-SELF--">
      <textarea name="brief" rows="5" cols="30">请输入简要说明</textarea>
</form>
```

显示结果如下：




### 5. button 按钮输入标记

按钮是最常使用的一种输入控件，一般形式是：

```
<input type="button" …>
```

主要属性有：

- name 属性：name 为按钮名称，便于程序对按钮的操作。
- value 属性：value 为按钮的显示名称。

button 按钮除了具有若干的一般属性外，同时还可以接受各种鼠标事件，具有不同的鼠标和键盘事件属性。例如，有如下 HTML 代码：

```
<form name="myForm" method="POST" action="/progs/feedback.jsp">
<input type="button" value="回前一页" onclick="history.go(-1);return true;">
<input type="button" value="关闭窗口" onclick="window.close();return true;">
</form>
```

显示结果如下：

回前一页 关闭窗口

当单击“回前一页”按钮的时候，则回到前面打开的页面，单击“关闭窗口”按钮的时候，当前窗口被关闭。

### 6. radio 单选钮输入标记

单选钮主要用于从多个可选项中选择其中之一。往往是若干个 radio 为一组，它们具有相同的 name，不同的 value，从而选择其中之一。

- name 属性：单选钮的名称，一般是若干个 radio 一组，取相同的 name。
- checked 属性：用来设置该单选框默认时是否被选中，相同 name 的多个 radio 中只能有一个选择，或都不使用该参数。
- value 属性：存储单选钮的取值，多个具有相同 name 的单选钮应该具有不同的 value。

例如，HTML代码如下：

```
<form name = "myForm" method = "POST" action = "/custom/feedback.jsp">
  性别：
  <input type = "radio" name = "gender" value = "Female">女性
  <input type = "radio" name = "gender" value = "Male" checked>男性
  <br><br>
  学历：
  <input type = "radio" name = "degree" value = "Bachelor" checked>学士
  <input type = "radio" name = "degree" value = "Master">硕士
  <input type = "radio" name = "degree" value = "Doctor">博士
</form>
```

上述代码的显示结果如下：

性别： ○女性 ◉男性

学历： ◉学士 ○硕士 ○博士

### 7. 复选框输入标记

复选框是对某种输入做出“是”或“否”的选择，一般形式是：

```
<input type = " checkbox " …>
```

和radio不同，每一个checkbox都是独立的。chebox的主要属性有：

- name属性：name为复选框的名称，便于程序获取用户输入。
- value属性：每一个checkbox必须有一个value，当复选框选中时，value值便会传到表单的action属性指定的程序中。
- checked属性：用来设置该复选框默认时是否被选中。

例如，HTML代码如下：

```
<form name = "myForm" method = "POST" action = "/progs/feedback.jsp">
     兴趣爱好：<br><br>
     <input type = "checkbox" name = "intrests01" value = "Sports" checked>体育
     <input type = "checkbox" name = "intrests02" value = "Music">音乐
     <input type = "checkbox" name = "intrests03" value = "Arts">文学
     <input type = "checkbox" name = "intrests04" value = "Others" checked>其他
</form>
```

显示结果如下：

兴趣爱好：

☑体育 ☐音乐 ☐文学 ☑其他

### 8. 复选列表框输入标记

创建一个下拉列表框或可以复选的列表框，定义复选列表框不需要在<input type=" ">中指定输入类型，其一般形式是：

```
<select name = "" size = "">
   <option value = "">…</option>
   <option value = "">…</option>
```

```
    …
</select>
```

1）<select>标记及其属性

- name 属性：name 为下拉式列表控件名称，便于程序获取用户输入。
- size 属性：下拉式列表的高度，默认时值为 1，若没有设置(加入)multiple 属性，显示的将是一个弹出式的列表框。若使用此参数则不会有 PopUp 效果。如果小于可选的项目数量，则出现垂直滚动条。
- multiple 属性：指定是否可以多选。multiple 属性不用赋值，直接加入<select>标记中即可使用，加入了此属性后列表框就成了可多选的了。

2）<option>…</option>标记

<option>标记用来指定列表框中的一个选项，主要属性有：

- value 属性：用来给<option>指定的那一个选项赋值，这个值将传送到服务器，服务器通过调用<select>标记的 name 的 value 属性来获得该区域选中的数据项。
- selected 属性：用来指定默认的选项，一个下拉式复选框可以有一项或零项被选中。

例如，HTML 代码如下：

```
<form name = "myForm" method = "POST" action = "/custom/feedback.jsp">
城市：
<select name = "city">
   <option value = "beijing">北京</option>
   <option value = "jinan" selected>济南</option>
   <option value = "qingdao">青岛</option>
</select>
</form>
```

显示结果为：

城市： 济南

如果指定 <select>的 size 属性，比如 size=“5”，则显示高度为 5 的一个列表框，不出现 PopUp 效果，占用较大的屏幕空间。

#### 9. Image 按钮标记

Image 输入类型，通常用于取代 submit/reset 两个默认的按钮，来显示个性化的按钮，Image 输入的一般形式是：

```
<input type = "image" …>
```

主要属性有：

- name 属性：所要代表的按键，可以是 submit、reset 或其他。
- src 属性：设置按钮图片，如果按键图片文件不与该 HTML 文件在同一目录下，需要加上正确的相对路径。

例如，HTML 代码如下：

```
<form name = "myForm" method = "POST" action = "/custom/feedback.jsp">
   <input type = "image" src = "/images/myOk.gif" name = "submit" width = "40" height = "40">
</form>
```

显示结果为：Ok

### 10. hidden隐藏元素标记

在一个表单中，可以定义隐藏表单元素，它在网页上并不显示，不需要用户输入，主要目的是随表单一起传给表单处理程序一个特定的值，用于为网页处理程序传送数据。

一般形式是：

```
<input type="hidden" name=" " value=" "…>
```

其中，name属性为控件名称，便于程序获取用户输入；value属性存储输入元素的默认值。例如，有如下代码：

```
<input type="hidden" name="myID" value="730118">
```

当表单提交后，服务器程序可以获得myID的值是730118，从而实现传送数据的目的。

### 11. 文件上传标记

为了实现通过HTTP协议上传文件，HTML规范在表单中增加了新的输入类型，文件上传标记的一般形式是：

```
<input type="file" name=" " size=" " accept=" " >
```

对于file类型输入，浏览器将显示一个文本框，用于文件名的输入，同时在文本框的后面显示一个“浏览…”按钮，允许用户通过浏览的方式选择要上传的文件。name属性为控件名称，size属性显示文本框长度。Accept属性设置上载文件过滤，即单击“浏览”按钮时，只显示指定文件类型的文件列表。

使用文件上传输入类型，在表单标记<form>中需要加入编码方案属性enctype=“multipart/form-data”。该编码方案在传送大量数据时比默认的表单编码方案“application/x-url-encoded”效率更高。因为URL编码只有很有限的字符集，当使用任何超出字符集的字符时，必须用“%nn”代替(nn表示两个十进制数)，因此，通过URL编码方式上载的文件大小将是原来的2～3倍。

例如，有如下代码：

```
<form name="myForm" method="POST" action="/custom/feedback.jsp"
      enctype="multipart/form-data">
  提交论文：<input type="file" name="F1" size="20">
</form>
```

显示结果为：

提交论文： 浏览...

用户单击“浏览”按钮，选择要提交的文件，文件将被上传到Web服务器。因为安全的原因，在HTML中，不能设置上传文件在服务器上的存储路径。上传文件的存储路径是在表单处理程序中设置的，在Web服务器端，通过组件，来设置每一个<input type="file">上传文件的存储路径。一个HTML表单可以设置多个<input type="file">控件，从而一次上传多个文件到Web服务器。

### 12. 表单提交按钮标记 submit/reset

最后，我们介绍两个最重要的按钮，就是当表单填写完毕后，需要选择的 submit/reset 按钮，它结束表单输入，将表单数据传送到服务器端，由表单<form>标记中的 action 属性指定的服务器上的程序处理用户输入数据。

1) 表单提交

表单提交按钮就是将表单内容提交给服务器，一般形式是：

```
<input type = "submit"…>
```

有如下属性：

- name 属性：这和其他控件的属性不同，在提交表单中，name 可以指定一个函数，需要和 form 标记中 action 属性的程序配合。一般情况下，不需要 name 属性。
- value 属性：提交按钮的显示名字，一般为“确定”、“提交”等易于理解的名字。

2) 重填按钮

表单清除就是要将表单中已做的输入和选择全部清除，重新填写。一般形式是：

```
<input type = "reset"…>
```

属性和表单提交相同。

下面是通过 FrontPage 在网页中插入表单的默认代码：

```
<form method = "POST" action = "--WEBBOT - SELF--">
   <!--webbot bot = "SaveResults" U - File = "fpweb:///_private/form_results.txt"
                                   S - Format = "TEXT/CSV" S - Label - Fields = "TRUE" -->
     <input type = "submit" value = "提交" name = "B1">
     <input type = "reset" value = "全部重写" name = "B2">
</form>
```

显示结果为：

提交 全部重写

在上述代码中，<form>标记中的 action 没有指定表单处理程序，则采用 Web 服务器端 FrontPage 扩展中一个默认的宏“--WEBBOT-SELF--”来处理用户的表单数据。这个 action 往往需要用户程序来代替，以便处理用户的表单输入。

**【例 3-5】** 一个利用 form 计算阶乘的例子。

```
<html>
<head>
<meta http - equiv = "Content - Type" content = "text/html; charset = gb2312">
<title>A simple example for Form</title>
<script language = "javascript">
var calcOnly = false;
function fact(n)
{
  if  (n == 0)
      return 1;
  else
      return n * fact(n - 1);
}
```

```
function calcFact(n)
{
  var res;
  res = fact(n);
  document.myForm.result.value = res;
  calcOnly = true;
  document.myForm.n.focus();
}
</script>
</head>
<body>
<p align = "center"><font size = "5">使用表单示例</font>
<hr>
<form name = "myForm">
<div align = "center">
   <table border = "1" cellpadding = "0" width = "303" height = "62">
      <tr>
         <td colspan = "3" height = "26" width = "292" bgcolor = "#C0C0C0">
            <p align = "center">计算 n 的阶乘</td>
      </tr>
      <tr>
         <td>输入 n<input type = "text" name = "n" size = "6" onChange = "calcFact(this.value)">
         </td>
         <td><input type = "button" value = "阶乘等于" name = "equ"
                        onClick = "calcFact(n.value)">
         </td>
         <td><input type = "text" name = "result" size = "13"
                        onChange = "if (calcOnly) { alert('This is a calculated field.');}">
         </td>
      </tr>
  </table>
</div>
</form>
</body>
</html>
```

网页在 Web 浏览器中的执行结果如图 3-4 所示。




图 3-4 表单在浏览器窗口中的显示

通过上面的例子,可以总结出以下几点:

(1) 不是所有的 form 表单都需要提交到 Web 服务器处理,即可以没有 action 属性。

(2) 如果是用 FrontPage 的 table 进行页面布局,在单元格中插入 form 表单控件时,生成的 HTML 代码比较混乱,甚至会出现多个 form,此时,要手工调整 HTML 代码,只要将所有的输入控件包含在<form>…</form>标记对之间就可以了。

### 3.2.7 层次块标记

为了更好地实现元素的精确定位,浏览器厂商在 HTML 规范基础上,在网页中增加了层(layer)和位置(position)的概念。通过层次块的显示、隐藏和移动来实现灵活的页面显示效果。不同的浏览器,对层的支持是不一样的,IE 浏览器支持<div>…</div>层次块标记。

层次块标记<div>…</div>用于定义网页上的一个矩形块,中间可以包含引起行中断的标记,如<table>标记等。层次块标记的一般形式是:

```
<div style = " " id = " ">
</div>
```

主要属性有:

- id 属性:用于标记一个<div>块,以便引用该块。
- style 属性:定义图层块的位置、大小、显示属性等。

此外,图层还可以接受 onclick 等鼠标事件,来增加交互功能。

在通常情况下,在网页上,我们可以用<div>标记和</div>标记来定义一个矩形区域,即定义一个图层块,通过图层块的 style 属性操作,来得到一些特殊的效果。通过客户端程序,可以实现对区域的显示、隐藏和移动等操作。如果没有层次块,要实现一个区域,例如一个 table 的移动是很不方便的。

**【例 3-6】** 定义 div 块并进行平滑移动。

在浏览器窗口定义一个 div 块,双击鼠标,div 块从左向右平滑移动,单击鼠标,停止移动。实现代码如下:

```
<html>
<head>
<title>Moving Div Sample</title>
<script language = "javascript">
var movingID = null;
var scrolling = false;
function startMove()
{
  var left = eval(div1.style.left.replace("px",""));
  if (left <document.body.scrollWidth - 150)
       div1.style.left = left + 1;
  else
       div1.style.left = 10;
  movingID = setTimeout("startMove()",10);
```

```
  }
  function stopMove()
  {
    clearTimeout(movingID);
  }
</script>
</head>
<body onDblClick = "startMove()" onMouseDown = "stopMove()">
    <div id = "div1" style = "visibility:visible; position:absolute; left:10; top:10; z - index:1;">
        <table bgColor = "#ffffcc" border = "1" cellPadding = "0" cellSpacing = "0">
            <tr>
               <td>Div moving....</td>
            </tr>
        </table>
     </div>
     <p>双击鼠标,块开始从左向右移动</p>
     <p>单击鼠标,块开始从左向右移动</p>
</body>
</html>
```

上述例子不仅是为了解释<div>标记,更重要的是要说明<body>标记的事件属性,这不可避免地用到了我们尚未解释的 JavaScript 脚本语言,由于可读性较好,相信读者能够看懂它的意思。也相信通过这个简单的例子,使大家初步体会 Web 应用的客户端编程特点和思路。

**【例 3-7】** HTML 标记综合举例。

下述代码演示了<a>标记、<div>标记以及<marquee>标记的组合应用,当鼠标指向一个超文本链接时,在鼠标的右下角显示 marquee 效果。

```
<html>
<head>
<meta http - equiv = "Content - Language" content = "zh - cn">
<title>Shandong Travel</title>
<script>
function showtip(current,e,text)
{   //IE 浏览器
   if (document.all && document.readyState == "complete"){
       document.all.MyTooltip.innerHTML
                   = "<marquee style = \"border:1px solid black\" >" + text + "</marquee>";
       document.all.MyTooltip.style.pixelLeft = event.clientX + document.body.scrollLeft + 10;
       document.all.MyTooltip.style.pixelTop = event.clientY + document.body.scrollTop + 10;
       document.all.MyTooltip.style.visibility = "visible";
   }
}
function hidetip()
{
  if (document.all)
      document.all.MyTooltip.style.visibility = "hidden"
  }
</script>
```

```
</head>
<body>
<a href = "http://www.sdta.gov.cn/" onMouseover = showtip(this,event,"大而强富而美的山东")
  onMouseout = "hidetip()">美丽的山东</a>
<div id = "MyTooltip"
      style = "position:absolute;visibility:hidden;clip:rect(0 150 50 0);
      width:150px;background - color:lightyellow">
</div>
</body>
</html>
```

### 3.2.8 对象和脚本程序标记

在 HTML 网页中,除了一些基本的文本、图片等内容外,还可以插入脚本程序、Java 程序、内嵌对象、Active 控件等,从而制作出更加新颖、丰富和交互能力更强的 Web 页面。这些新的文档内容需要通过<object>、<script>、<applet>、<bssound>、<embed>等标记来进行标记。下面介绍其中的插入对象标记和脚本程序定义标记。

#### 1. 插入对象标记<object>…</object>

插入对象包括 Flash 动画、ActiveX 组件或其他对象。在 HTML 中插入一个对象,需要使用<object>…</object>标记来标记对象,在<object>标记内,还需要使用多个<param>标记,来设置该对象属性的初始值,即为该对象传递参数值。一般形式如下:

```
<object classid = "" id = "obj1">
    <param >
    <param >
    ⋮
</object>
```

通常情况下,<object>标记以及每一个子标记<param>都包含了许多属性,插入对象的类型不同,这些参数的设置悬殊较大。例如,插入一个 Flash 对象对应的标记是:

```
    <object classid = "clsid:D27CDB6E-AE6D-11CF-96B8-444553540000" id = "obj1"
codebase = "http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab # version =
6,0,40,0" border = "0" width = "640" height = "480">
        <param name = "movie" value = "/images/csmacd.swf">
        <param name = "quality" value = "High">
        <embed src = "/images/csmacd.swf"
pluginspage = "http://www.macromedia.com/go/getflashplayer"
type = "application/x-shockwave-flash" name = "obj1" width = "640" height = "480">
    </object>
```

对于各种类型对象的详细介绍略。

#### 2. 脚本程序定义标记<script>…</script>

用户可以在 HTML 文件中插入脚本程序,脚本程序分为客户端和服务端脚本程序两

类。客户端脚本程序一般为 JavaScript 程序，在客户端浏览器中解释执行，可以在文件头和文档体内书写脚本程序。服务端脚本程序与 Web 服务器有关，服务端脚本通常是在文档体内定义。不同的 Web 服务器应该对应不同的服务端脚本程序，因为服务端脚本程序是在 Web 服务器上运行的。

在 HTML 文件中标记脚本程序的一般形式是：

```
<script language = "" runat = "" >
      脚本程序代码
</script>
```

属性 language 用于设置脚本程序语言，runat 属性设置脚本是客户端脚本还是服务端脚本，默认为客户端脚本。要声明是服务端脚本，则属性值设为 sunat＝"Server"。服务端脚本程序也可以使用其他的标记，如 Tomcat 支持的服务器脚本标记为＜％…％＞标记。

### 3.2.9　层叠样式表 CSS 技术

在 HTML 中，每一个标记都包含了默认的显示样式，定义了所标记内容在浏览器中的布局和显示外观。如果要修改一个标记的默认显示属性，需要通过标记的 style 属性为标记的相应属性赋值，这使得标记更加复杂，难以维护。W3C 为了解决 HTML 的结构化问题和实现 Web 中的总体外观控制，于 1996 年底，公布了层叠样式表单(Cascading Style Sheet，CSS)规范。所谓层叠是指对于容器元素指定的所有选项，将被自动地应用到其包含的所有元素中。

在 CSS 规范中，可以定义标记的样式类，或者为特定的标记定义 ID 属性，然后在标记中，将这些标记属性存储在 HTML 头部的＜style＞…＜/style＞中，或单独保存为.css 文件，然后通过标记的 class 属性或 ID 属性来引用，从而实现标记内容的定制显示外观。

#### 1. 标记的 style 属性与内联样式

在 HTML 中，除了 basefont、param 和 script 标记外，几乎所有的文档体内的标记都有 style 属性，包括＜body＞标记本身。style 属性将修改标记的默认样式，从而实现标记按照特定的形式显示，这就是内联样式。例如：

```
<p style = "color: red; font - family:'Impact'">红色英文 Imapct 字，如果字体可用的话。</p>
```

要使用内联样式，必须在文档的＜head＞…＜/head＞部分包括以下标记：

```
<meta http - equiv = "Content - Type" content = "text/css">
```

内联样式可以改变标记的默认显示样式，如果需要多个标记使用同样的样式显示时，需要为每一个标记添加 style 属性，非常麻烦。另外，内联样式和需要展示的内容混合在一起，显得比较混乱，修改不够方便，这本身也是 HTML 规范的一大弊端。

总之，style 属性可以改变标记的默认显示样式，采用 style 属性管理大量文档的显示将十分困难。当管理人员需要改变某些标记的显示形式时，需要做大量的微调工作。要解决站点级的显示问题，内联样式将非常困难，必须借助于样式表。

### 2. 修改标记默认样式

当一个用户自定义的样式要应用到文档中的所有元素(标记)时,应该在文档的头部使用<style>…</style>标记对重新定义这些标记的样式。

<style>标记放置在 HTML 文档的<head>…</head>内,用于定义样式。使用<style>标记可以为网页设置不同的样式属性,一般形式为:

```
<style type="text/css">
    标记 { 属性名: 属性值; 属性名: 属性值; }
</style>
```

在<style>和</style>之间的部分是一系列的样式定义,实际上,就是对 HTML 标记的默认显示样式进行重新定义,做到个性化。要修改一个标记的默认属性,需要知道这个标记有哪些属性,记住一个标记的所有属性是很困难的,这需要借助于 FrontPage、Dreamweaver 等网页制作工具来定义样式。例如,在 FrontPager 的菜单"格式"中,包含"样式…"命令,可以修改标记的默认样式,或者新建样式类、创建 CSS 文件等。

例如,要设置整个文档的文字颜色和背景色,可以定义样式为:

```
<style type="text/css">
   body { color: black; background: white; }
</style>
```

例如,要个性化超链接的显示,可以定义下面的样式:

```
<style type="text/css">
  a:hover { color:#FF0000;
            text-decoration:none;
            font-weight:bold}
  a {color:#0000FF;text-decoration:none; font-size:14px}
</style>
```

这样,对于文档中的超链接(<a>…</a>),文字显示为蓝色,当鼠标指向的时候则显示红色,同时字体也为 14px,不显示下划线。

### 3. 标记的 id 属性与 class 属性

如果将一个样式用到某个元素的一个场合,而不是该标记的全部,此时可以在文件头部的<style>…</style>标记内定义通用 id 值或隶属于某个标记的 id 值,或者定义样式类,然后在标记中设置 id 属性和 class 属性。

元素样式定义的语法格式为:

```
tagName[.className][#ID 属性名] {
    property : value
    [;property : value....]
}
```

在 HTML 中,可以设置的 CSS 属性很多,不同的标记,可设置的 CSS 属性也不相同。在具体应用中,不需要记忆众多的 CSS 属性,所有的网页制作工具,例如 FrontPage、

Dreamweaver 等，都具有定义 CSS 的功能，并且是所见即所得。

(1) ID 属性用于定义一个元素的独特的样式。ID 值可以关联一个标准的标记，也可以用于任何标记。例如：

```
#myID1  { font-size: larger }
p#myNote { font-weight: bolder; color: red; background: white }
```

上述的 myID1 不隶属于任何标记，因此可以在相关的标记中设置 id 属性为 myID1；myNote 隶属于段落标记<p>，因此，只能在标记<p>中使用。例如：

```
<div id=myID1>欢迎使用 ID 属性</div>
<p id=myNote>注意事项：</p>
```

当一个样式只需要在任何文档中应用一次时，使用 id 是很适合的。

(2) class 属性用于指定标记使用的样式类。样式类可以关联一个标准的标记，也可以用于任何标记。例如：

```
.word1 { color: lime; background: #ff80c0 }
p.warning { font-weight: bolder; color: red; background: white }
```

其中，word1 类不隶属于任何标记，可以用于任何 body 元素，因为它在样式表中没有和具体的 HTML 元素关联。但 warning 类只能用于段落标记<p>。

当定义了样式类后，可以在标记中通过 class 属性引用，例如：

```
<p class=warning>警告：</p>
<p class=word1>Please turn off the power first</p>
```

#### 4. 样式(.css)文件

在 HTML 文档的<head>…</head>标记内，通过<style>…</style>定义的样式 ID 和样式类，只能应用于当前的 HTML 文档。如果要将这些样式应用到其他 HTML 文档中，应该使用样式文件。即将这些样式定义存储在一个扩展名为 css 的样式文件中，css 文件可以是一个标准的 HTML 文件，只不过<body>…</body>为空。然后，当某个网页需要使用其中的样式时，在文档的<head>…</head>中增加<link>标记，一般形式如下：

```
<link type="text/css" rel="stylesheet" href="mystyle.css">
```

这样，在当前文档中，就可以使用样式文件 mystyle.css 中定义的样式了。

一种良好的 HTML 页面就是充分利用 css 技术，将页面的显示和布局分开，从而保证页面维护的灵活性。例如，我们要设计一个登录界面 login.htm，如图 3-5 所示。

上述登录界面可以用表格来实现，对于表格，为了增加修改的灵活性，可以定义一个该页面的 css 文件，例如 login.css。因此，登录界面就需要两个文件来构成，一个是 login.html，另一个是所用到的 css 文件 login.css，存储在 styles 文件夹中。

登录页面文件 login.html 代码清单如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-transitional.dtd">
```

图 3-5 用户登录界面

```
<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
<meta http - equiv = "Content - Type" content = "text/html; charset = gb2312" />
<link rel = "stylesheet" type = "text/css" href = "styles/login.css" />
</head>

<body>
<form action = "login.jsp" method = "post" onSubmit = "return on_submit()" name = "form1">
<table id = "login" cellpadding = "0" cellspacing = "0">
<tr>
    <td id = "login_row1">用户登录</td>
</tr>
<tr>
    <td id = "login_row2">用户名：<input class = "input" type = "text" name = "username"
value = ""/></td>
</tr>
<tr>
    <td id = "login_row2">密   码：<input class = "input" type = "password" name =
"username" value = ""/></td>
</tr>
<tr>
    <td class = "login_submit"><input type = "submit" value = "登录"
class = "mybutton">    <input type = "reset" value = "重置"
class = "mybutton"></td>
</tr>
</table>
</form>
</body>
</html>
```

在上述 HTML 代码中，对于每一个标记，其显示样式使用了 id 属性和 class 属性，这些属性在 login.css 中进行了定义。login.css 代码清单如下：

```
body{
    margin:0 auto;
    font - size:12px;
```

```
    text-align:center;
    background-color:#FFFFFF;
    font-family:宋体,Helvetica,sans-serif;
}
#login{
      width:500px;
      height:300px;
      background:#CCCCCC;
      margin-top:100px;
      border:#000000 1px solid;
}
#login_row1{
      height:100px;
      font-size:36px;
      color:#000000;
      text-align:center;
      vertical-align:middle;
}
#login_row2{
      height:50px;
      color:#000000;
      text-align:center;
      vertical-align:middle;
}
.login_submit{
     height:50px;
     color:#000000;
     text-align:center;
     vertical-align:middle;
}
.input{
     border-right:1px solid #0163A2;
     border-bottom:1px solid #0163A2;
     border-left:1px solid #0163A2;
     border-top:1px solid #0163A2;
     height:18px;
     width:200px;
     margin-left:5px;
     background:#FFFFFF;
     text-align:left;
     vertical-align:middle;
}
.mybutton{
    height:20px;
    width:50px;
    border-right:1px solid #0163A2;
    border-bottom:1px solid #0163A2;
    border-left:1px solid #0163A2;
    border-top:1px solid #0163A2;
    font-size:12px;
    color:#0163A2;
```

```
    text-align:center;
    vertical-align:middle;
    background:#FFFFFF;
}
```

需要特别说明的是，在上述的 css 文件定义中，如果一种样式在 HTML 中需要通过 id 属性来引用，应使用“#”符号来定义，如果要通过 class 来使用，则应通过“.”符号来定义。对于一些常用的 css，可以分别存储在不同的.css 文件中，然后将这些.css 文件保存在一个 styles 文件夹中。

### 3.2.10 帧

帧(frame)可以用来将浏览器窗口划分为多个区域(子窗口)，每个子窗口中装载一个 HTML 文件。即每个 HTML 文件占据一个帧，而多个帧可以同时显示在同一个浏览器窗口中，这样的 Web 页面称为框架网页。

帧页定义的一般形式是：

```
<frameset rows="" cols="">
  <frame name="" target="" src="">
  <frame name="" target="" src="">
  …
</frameset>
```

其中<frameset>…</frameset>标记定义帧，必须有一个 rows 属性或 cols 属性，将屏幕分成若干行或若干列。然后跟着是相应的每一帧定义，由<frame>来标记，<frame>为单标记。如果某个<frame>进一步进行了拆分，在<frame>处，可以嵌套<frameset>…</frameset>。

#### 1. <frameset>…</frameset> 标记

<frameset>…</frameset>标记对放在帧的主文档的<body></body>标记对的外边，也可以嵌在其他帧文档中，并且可以嵌套使用。此标记对用来定义主文档中有几个帧并且各个帧是如何排列的。它具有 rows 和 cols 属性，使用<frameset>标记时这两个属性必须至少选择一个，否则浏览器只显示第一个定义的帧。

rows 用来规定主文档中各个帧的行定位，而 cols 用来规定主文档中各个帧的列定位。这两个属性的取值可以是百分数、绝对像素值或星号(“*”)，其中星号代表那些未被说明的空间，如果同一个属性中出现多个星号则将剩下的未被说明的空间平均分配。同时，所有的帧按照 rows 和 cols 的值从左到右，从上到下排列。

例如，通过 FrontPage 2003，新建“标题、页脚和目录”框架网页，代码如下：

```
<html>
<head>
<meta http-equiv="Content-Type" CONTENT="text/html; charset=gb2312">
<meta name="GENERATOR" content="Microsoft FrontPage 4.0">
<meta name="ProgId" content="FrontPage.Editor.Document">
<title>New Page 1</title>
```

```
</head>
<frameset rows = "64, * ,64">
    <frame name = "top" scrolling = "no" noresize target = "contents" src = "new_page_2.htm">
    <frameset cols = "150, * ">
        <frame name = "contents" target = "main" src = "new_page_3.htm">
        <frame name = "main" src = "new_page_4.htm">
    </frameset>
    <frame name = "bottom" scrolling = "no" noresize target = "contents" src = "new_page_5.htm">
    <noframes>
    <body>
    <p>此网页使用了框架,但您的浏览器不支持框架。</p>
    </body>
    </noframes>
</frameset>
</html>
```

上述代码将屏幕分成了三行,其中第二行又分成两列,一共四个帧,每个帧对应一个文件,对应<frame>标记中的 src 属性,另外一个文件存储上述的代码,即框架网页,或称为主帧网页。显示结果如图 3-6 所示。

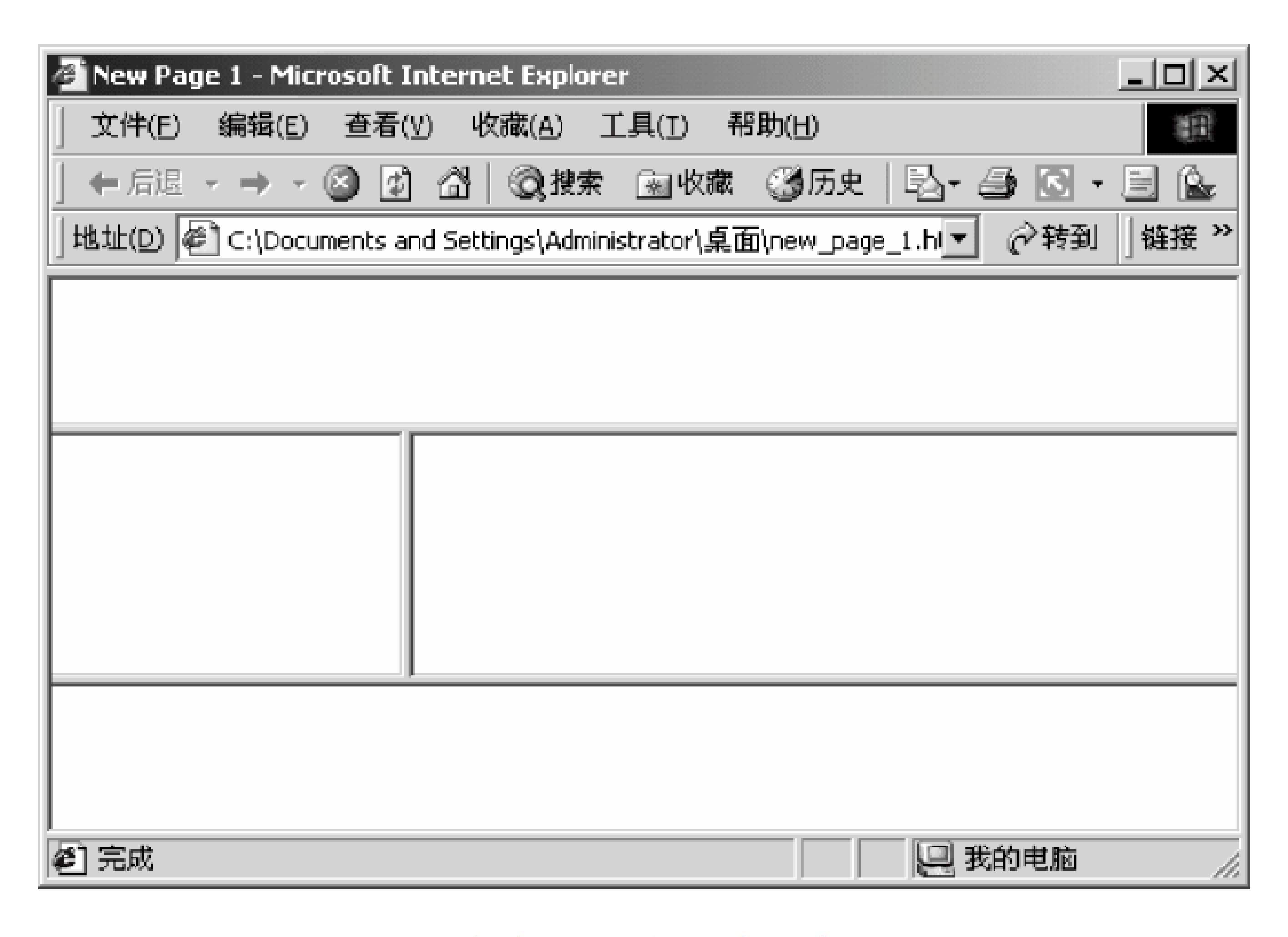



图 3-6 框架网页在浏览器中的显示

### 2. <frame> 标记

<frame>标记放在<frameset>…</frameset>之间,用来定义一个具体的帧。<frame>标记具有 src 和 name 属性,这两个属性都是必须赋值的。src 是此帧的源 HTML 文件名(包括网络路径,即相对路径或网址),浏览器将会在此帧中显示 src 指定的 HTML 文件;name 是此帧的名字,这个名字是用来供超文本链接标志<a href="" target="">中的 target 属性指定链接的 HTML 文件将显示在哪一个帧中。

例如,定义了一个帧,名字是 main,在帧中要显示的 HTML 文件为 mint.htm,则代码是<frame name="main" src="mint.htm">。如果有一个链接,在点击了该链接后,需要在 main 的帧中显示文件 newone.htm,则代码为<a href="newone.htm" target="main">链

接的文本</a>。这样一来,就可以在一个帧中建立网站的目录,加入一系列链接,当点击链接以后在另一个帧中显示被链接的 HTML 文件。

此外,<frame>标记还有 scrolling 和 noresize 属性,scrolling 用来指定是否显示滚动条,取值可以是“yes”(显示)、“no”(不显示)或“auto”(若需要则会自动显示,不需要则不显示)。noresize 属性直接加入标记中,不需赋值,它用来禁止用户调整一个帧的大小。

#### 3. <noframes>…</noframes> 标记

有的浏览器不支持框架网页,此时需要使用<noframes>…</noframes>标记对,用来在那些不支持帧的浏览器中书写传统的<body>…</body>部分。

## 3.3 扩展标记语言 XML

在互联网的发展历史上,有两种非常核心的技术,这就是 Java 和 XML。Java 提供了程序代码的平台无关性,而 XML 则保证了数据的平台无关性,已成为 Web 应用中数据表示和数据交换的标准。但是,人们对 XML 的认识远远没有对 HTML 的认识彻底和清晰,有些理解甚至是完全错误的。那么,究竟什么是 XML 呢? XML 和 HTML 有什么不同,它们的本质区别是什么呢? 同时,由于 XML 毋庸置疑的优越性以及 XML 的不断发展壮大,挂在“XML”一词下的标准和规范不断变化,了解这些标准的来龙去脉,以及它们之间的关系,对于掌握 XML 也是至关重要的。

### 3.3.1 XML 技术简介

XML 和 HTML 都称为标记语言,但是两者有着本质的不同。HTML 的目的是标记内容的显示样式,但是,XML 的基本动机则是对数据结构的表达,实现内容和内容展示的分离,要显示 XML 文档内容,则需要其他相应的规范,例如 XSLT 等。

#### 1. HTML 的不足

HTML 的出现无疑是 Internet 技术和 Web 技术的一次突破,为推动 Internet 和 Web 技术的发展发挥了巨大的作用。随着 Web 技术的快速发展,Internet 上的 Web 信息越来越多,内容越来越复杂,数据格式也越来越多,HTML 已经无法满足表达日益丰富的数据形式的需要。HTML 的不足主要表现在以下几个方面:

(1) HTML 的标记固定,HTML 只是一种表现技术,不能表达语义。

(2) 不能适应现在越来越多的网络设备和应用的需要,比如手机、PDA、信息家电都不能直接显示 HTML。

(3) HTML 代码不规范、臃肿,浏览器需要足够智能和庞大才能够正确显示 HTML。

(4) 数据与表现混杂,页面要改变显示,就必须重新制作 HTML。

#### 2. XML 的产生和发展

HTML 的不足推动了 XML 的产生和发展。1996 年 8 月,那些关心 SGML 的专家聚

集在美国西雅图，成立了一个名为GCA(Graphic Communications Association，图形通信协会)的组织，研究如何开发SGML以便它适应和促进Web技术的发展。他们对SGML过于复杂难于被理解和实现的方面进行简化，去掉其语法定义部分，适当简化DTD部分，并增加了部分互联网的特殊成分。为了体现它与HTML的不同，工作组将其命名为XML(eXtensible Markup Language)，同时也将自身更名为XML工作组。1998年2月10日，XML工作组正式向W3C提交了XML的最终推荐标准，这就是XML 1.0标准。

在XML中，SGML的最初动机得以延续，那就是将文件内容和处理这些内容的应用程序进行分离，在文件内容中不嵌入数据的处理过程代码，文件内容被编码为条理清晰的文本，从而便于数据交换和处理。对数据进行研究有着重要的意义，因为数据往往是相对稳定的，变化的通常是处理这些数据的程序。实现数据和操作这些数据的程序的分离是XML的设计动机，这是深刻理解XML的基础。在XML中，如果XML某方面设计的与应用程序太过紧密，就可以认为这是一种bug，这是使用XML最重要的一个原则。

XML标准的发展没有HTML那样迅速，直到2002年10月15日，W3C才发布了XML 1.1候选推荐标准。在XML 1.0规范中，使用的字符集为Unicode 2.0。随着Unicode版本的升级，XML 1.1支持新的Unicode字符，不再局限于一个具体的Unicode版本。此外，在XML 1.1中，增加了IBM大型主机规定的换行符(＃x85：十六进制的85)和Unicode换行符(＃x2028)的处理能力，这些改变都提高了XML的国际化支持水平。

#### 3. XML分析器

XML文档需要在XML分析器中才能运行，XML分析器是一个软件模块，应用程序利用它来解析XML文档并且得以访问数据和数据结构。XML分析器有确认型和非确认型两种。确认型XML文档分析器检查XML文档的语法，将XML文档内容同文档类型定义DTD和架构作比较。XML分析器通过判断XML数据是否和预定义的确认规则相符，以判定XML文档是否为构造良好。非确认型XML分析器也进行XML文档语法的检查，但不进行XML文档内容和DTD及架构的比较。

现在几乎所有的主流浏览器都内置了XML分析器，支持XML和XSLT。例如，在微软的Internet Explorer浏览器中，内置了XML确定性分析器MS XML。如果是一个有效的XML文件，在浏览器中打开XML文件时，XML文档将显示为一个树状结构，称为XML文档树。通过点击元素左侧的加号或减号，可以展开或折叠元素结构。此外，浏览器或其他应用程序也可以使用CSS或XSLT样式转换显示XML文档数据。

#### 4. XML技术分析

随着Internet的快速发展，尤其是电子商务、Web服务等应用的广泛使用，XML类型的数据成为当前主流的数据形式。虽然XML作为一种通用的数据交换语言，已经成为业界的一种具有垄断性的标准，在跨平台跨系统数据交换方面拥有无可比拟的优势，但是，和关系数据库相比，XML的最大缺陷就是它的效率较低。因为在关系型数据库中，数据的字段名只需要出现一次，但是在XML数据文件中，元素名将反复出现，这必然会影响到查询的效率。

作为一种数据存储方案，XML技术无疑具有绝对的优势。提高数据的查询、使用效率

成为 XML 技术研究的热点,这些研究包括 XML 与关系数据库之间的相互转换,利用关系数据库的成熟技术对 XML 数据进行处理。为提高 XML 的查询效率,需要为 XML 类型提供索引功能。如果对 XML 文档不构建索引结构,那么针对 XML 数据的任何查询都很可能导致对整个文档树的遍历,随着 XML 数据集的增大,这种开销是不可忍受的。

上述关于 XML 研究和应用的需求推动了 XML 的发展,也导致了许多新的 XML 相关标准和规范的产生,这包括:XML 架构、扩展样式语言 XSLT、XML 路径语言 XPath、XML 查询语言 Xquery、XML 链接语言规范 XLink 和 XPointer 等。

### 3.3.2 XML 文档结构

XML 文档是一个纯文本文件,XML 规范定义了 XML 文档良好的结构,将一个 XML 文档分为 XML 文档序言、文档类型定义和文档内容三个部分,一般形式如下:

```
<?xml version="1.0" encoding="GB2312" ?>                        ┐
<?xml-stylesheet type="text/xsl" href="myxslfile.xsl"?>          │ XML文档序言
<!DOCTYPE rootElementName SYSTEM "mydtdfile.dtd">                ┘

<!DOCTYPE rootElementName [                                      ┐
   <!ELEMENT element-name(elementdefination)>                    │ 文档类型定义
   ...                                                           │ (用户自定义标记)
]>                                                               ┘

<rootElementName>                                                                                   ┐
   <elementName1 [属性名 1="属性值"] [属性名 2="属性值"] ...>elementValue</elementName1>            │
   <elementName2 [属性名 1="属性值"] [属性名 2="属性值"] ...>elementValue</elementName2>            │ XML文档内容
        ⋮                                                                                           │
   <elementNamen [属性名 1="属性值"] [属性名 2="属性值"] ...>elementValue</elementNamen>            │
</rootElementName>                                                                                  ┘
```

#### 1. XML 文档序言

XML 文档序言是在文档类型定义和根元素的开始标记之前的部分,声明应用于整个文档的信息,包括 XML 文档声明(XML 版本、字符编码)、XML 样式转换语言文档以及外部 DTD 定义等。例如:

```
<?xml version = "1.0" encoding = "GB2312" ?>
<?xml - stylesheet type = "text/xsl" href = "myxslfile.xsl"?>
<! DOCTYPE rootElementName SYSTEM "mydtdfile.dtd">
```

在上述 XML 序言中,第一行为 XML 文档声明,声明 XML 文档为 XML 1.0,使用字符集为 GB2312。XML 声明通常在 XML 文档的第一行出现。XML 声明不是必选项,但是如果使用 XML 声明,必须在文档的第一行,前面不得包含任何其他内容或空白。

XML 文档声明的一般形式为:

```
<?xml version = "versionNumber"  [encoding = "encodingValue"] [standalone = "yes/no"] ?>
```

VersionNumber 为版本声明,声明 XML 文档所遵循的 XML 规范的版本号。encoding

为编码声明标识,用于表示文档中的字符的编码,为可选项,默认值为 UTF-8(适合美国英语的 Unicode 压缩版)。XML 分析器可以自动确定文档使用的是 UTF-8 还是 UTF-16 Unicode 编码,但是,在支持其他编码的文档中应使用此声明。独立声明参数 standalone 标识该文档是否依赖于其他 XML 文档,取值为 yes 或 no,默认值为 yes。

第二行为文档处理程序声明,声明处理该 XML 文档数据的外部程序,通常是一个 XSLT。

第三行为外部文档类型定义声明,即声明一个外部 DTD 文件,一般形式是:

```
<!DOCTYPE rootElementName SYSTEM "dtdFile">
```

在外部文档类型定义后面将是内部文档类型定义部分,可以省略。

### 2. 文档类型定义

在 XML 文档序言和 XML 文档之间的部分是文档类型定义(DTD)部分,用于定义 XML 中用到的元素、元素属性等,即声明用户自定义标记及相关属性,其目的是用于确认型 XML 分析器检查 XML 文档是否结构良好。相对于用外部的 DTD 声明,这里的文档类型定义称为内部 DTD 声明,其一般形式是:

```
<!DOCTYPE rootElementName [
    <!ELEMENT element-name(elementdefination)>
    ⋮
]>
```

其中,rootElementName 为文档的根元素,在根元素内定义其他元素。例如,<!ELEMENT address(buildingnumber, street, city, state, zip)>,则定义一个名称为 address 的元素,address 元素按顺序又包含<buildingnumber>、<street>、<city>、<state>、<zip> 五个元素。可以进一步定义<buildingnumber>元素为<!ELEMENT buildingnumber(#PCDATA)>,它确定了<buildingnumber>元素的取值。

在一个 XML 文档中,内部文档类型定义(DTD)部分不是必需的,如不需要可以省略。

### 3. XML 文档内容

XML 文档内容,即 XML 文档体,是 XML 文档的数据部分。文档体包括一个或多个元素,每个元素有一个开始标记和一个结束标记定义。文档体中的元素定义了数据的结构,有一个单独的根元素包含所有其他元素,文档中所有数据都包含在文档体的根元素中。

在文档体内,还可以使用名称空间,即在每一个元素和属性前面加上前缀“名称空间:”来唯一地标识一个元素或一组元素的属性,以避免多个 XML 文档中的元素重名。

**【例 3-8】** 一个简单的 XML 文档。

下面是 Brion 给 Jane 的便条,使用 XML 格式,内容如图 3-7 所示。

用 HTML 的思想,双击该文档在浏览器中打开,显示如图 3-8 所示。

在浏览器中,我们看到了一棵 XML 文档树。这个 XML 文档做了什么呢?好像什么都没做。我们需要的是观念上的转变,不能再以 HTML 的思想来理解 XML 了。XML 不是 HTML 的替代品,HTML 设计的目的是用来显示数据,重点是显示数据以及如何使数据的




图 3-7 一个简单的 XML 文档




图 3-8 XML 文档在浏览器中的显示界面

显示更美观。XML 是用来存储数据的,XML 设计的目的是用来描述数据结构,以及存储数据,实现数据存储和显示的分离,数据的显示则是通过层叠样式表 CSS 或样式转换语言 XSL 实现的。

### 3.3.3 文档类型定义 DTD

在 XML 中,没有像 HTML 一样拥有一个通用的标记集合,标记(在 XML 中,又称“元素”)是通过文档类型定义(Document Type Difinition,DTD)来实现的。DTD 定义了 XML 文档中可以使用的标记符号、标记的属性、标记的排列方式/顺序、标记能够包含的内容等,其目的是保证确认型 XML 分析器来确定 XML 文档数据的合理性,保证 XML 文档结构良好。

DTD 可以在 XML 文件中直接定义,也可以保存为一个完全独立的文件(.dtd)。因此,DTD 分为内部 DTD(在 XML 文件中直接定义 DTD)和外部 DTD(在 XML 文件中调用已经编辑好的 DTD 文件)两种。例如,对于一家企业,有自己的供应商、客户、合作伙伴,他们相互之间的交换电子文档都是用 XML 文档,那么我们可以将这些 XML 文档的 DTD 保存为一个独立的 DTD 文件,让所有要交换的 XML 文档都使用此 DTD。

#### 1. 在 DTD 中声明 XML 元素

一个内部 DTD 声明必须写在 XML 序言和 XML 根元素之间，一般形式为：

```
＜！DOCTYPE rootElementName[
    ＜！ELEMENT element－name (element－definition) ＞
    ⋮
]＞
```

其中，＜！DOCTYPE 表示开始设定 DTD。rootElementName 指定此 DTD 的根元素的名称，一个 XML 文件只能有一个根元素。

＜！ELEMENT element-name(element-definition)＞为元素定义语句，其中，＜！ELEMENT 是 XML 的保留字，表示开始元素定义。element-name 是为元素所起的名称，element-definition 是对元素的定义，就是说＜元素＞…＜/元素＞之间能够包含什么内容。元素的内容可以是一般性文字，也可以是其他元素。

element-definition 可以是：

```
EMPTY | ＃PCDATA | 元素 | ANY
```

- EMPTY：没有内容的元素。在 XML 文件中，空元素不需要结束标记，但必须采用＜/空元素名＞这样的写法。
- ＃PCDATA：声明一个基本元素，元素值为可解析字符数据。例如，＜！ELEMENT 名称(＃PCDATA)＞则定义一个元素“名称”，它由＃PCDATA 关键字定义，表明此元素仅仅包含一般文字，是基本元素。在文档内容中，可书写下面的 XML 内容：

```
＜名称＞The 6th International Conference on Web-based Learning＜/名称＞
```

- 元素：声明一个容器元素，即元素还可以包含另外的元素，形成一种嵌套和层次结构。
- ANY：表明所有可能的元素以及可解析的数据。

如果在一个元素定义中又包含其他元素，这样的元素为容器元素。声明容器元素的基本语法为：

```
＜！ELEMENT containerElement(containedElement1,…,containedElementn)＞
```

其中，containerElement 为容器元素名称，$containedElement_1$～$containedElement_n$ 为被包含的元素。被包含元素可以取下列三种格式之一：

- element，要求该元素有且只有一个值。
- element＋，要求该元素有一个或多个值。
- element＊，要求该元素有零个或多个值。

例如，＜！ELEMENT 书籍(名称，作者，价格)＞表示定义了一个容器元素(即标记)“书籍”，包含三个子元素，分别是“名称”、“作者”和“价格”。

在一些容器元素的声明中，有可能它包含的子元素是在多个子元素中的一个，那么在声明此父元素时，就可以把它声明成选择性元素，可供选择的子元素用“|”分隔。例如：＜！ELEMENT 配偶(妻子|丈夫)＞。

### 2. 在 DTD 中声明元素属性

和 HTML 标记一样，XML 元素往往也包含不同的属性。在 XML 中，元素属性设置的一般形式是：

```
<!ATTLIST element-name attribute-name Type Default-value>
```

其中，<!ATTLIST 表示开始属性的设定；element-name 表示元素名；attribute-name 是元素属性名称；Type 是该属性属性值的类别，元素属性值类型见表 3-4。

**表 3-4 XML 中元素属性类型列表**

| 属性类型 | 描述 |
|---|---|
| CDATA | 属性值为一般文字 |
| ENUMERATED | 枚举该属性的取值范围，一次只能有一个属性值能够赋予属性 |
| NMTOKEN | 表示属性值只能由字母、数字、下划线等符号组成 |
| NMTOKENS | 表示属性值能够由多个 nmtoken 组成，每个 nmtoken 之间用空格隔开 |
| ID | 该属性在 XML 文件中是唯一的，常用来表示人的身份证号码 |
| IDREF | 表示该属性值是参考了另一个 id 属性 |
| IDREFS | 表示该属性值是参考了多个 id 属性，这些 id 属性的值用空格隔开 |
| ENTITY | 表示该属性的设定值是一个外部的 entity，如一个图片文件 |
| ENTITIES | 该属性值包含了多个外部 entity，不同的 entity 之间用空格隔开 |
| NOTATION | 属性值是在 dtd 中声明过的 notation（声明用什么应用软件解读某些二进制文件，如图片文件） |

Default-value 是指该属性值的取值特点，有四种不同的属性取值，分别是：

- #REQUIRED：表示在标记中必须给定属性值。
- #IMPLIED：表示该属性值可以省略。
- #FIXED：表示一个固定的属性值。
- 字符串：指定属性的默认取值。

下面是一组 XML 元素和元素属性声明：

```
<!ELEMENT FAMILY (PERSON+)>
<!ELEMENT PERSON EMPTY>
<ATTLIST PERSON
        myID     ID #REQUIRED
        name     CDATA #REQUIRED
        sex      (male|female) "male"
        nickname NMTOKENS #IMPLIED
        parentID IDREFS #IMPLIED
>
```

上述 XML 语句为元素“PERSON”声明了“myID”、“name”、“sex”、“nickname”和“parentID”五个属性。属性 myID 属性类别为 ID，表明 myID 属性的取值在此 XML 文件中是唯一的，否则将出现解析错误。此属性设定中的属性取值要求为#REQUIRED，表示“myID”属性在元素“PERSON”中必须出现，否则也会产生解析错误。

属性 name 为 CDATA 属性类别，表明取值为一般性文字。属性 sex 的属性值类别是

枚举类型，取值范围为“male”或者“female”，如果在XML文件中没有为此属性赋值，属性默认取值是一个字符串“male”。

属性nickname属性类型为NMTOKENS，规定了其取值的字符集，此属性可以省略。属性parentID的类型为IDREFS，表明该属性的值必须在文档中出现过，该属性可以省略。如果该属性的值没在文档中出现过，解析器将认为该文档为不规范文档。

根据上面的元素属性说明，我们看下面的XML文档数据。

```
<FAMILY>
<PERSON myID="P_1" name="Brion" nickname="sun@#$">
<PERSON myID="P_2" name="Jane" sex="female">
<PERSON myID="P_3" name="Linda" sex="female">
<PERSON myID="P_4" parentID="P_1 P_5" name="David">
</FAMILY>
```

上述文档数据是不正确的，因为在第一个元素“PERSON” nickname属性值中包含了NMTOKENS所不允许的字符“@#$”。此外，parentID属性值中出现了值“P_5”，但该值没有在文档中出现过。

#### 3. 定义实体

在XML的DTD中，还可以定义实体(Entity)。实体实际上起一种类似“宏”的作用，一些常用的或者不便于直接书写的文字或数据，可以用一个标识定义下来，在数据中可以直接引用，这就是实体。实体的引用通过“&”来引用，末尾加“;”。

在XML中，有5种预定义实体，分别是：字符“&”(&)、“<”(<)、“>”(>)、“"”(")和“'”(')。除了这些预定义实体，还允许用户自己定义实体，例如，如果在XML文档中需要频繁使用词组“Good Luck”，可以在DTD中这样表示：<!ENTITY gl "Good Luck">。这样当使用这个词组“Good Luck”时，可以敲入≷，从而可以避免拼错和重复敲入相同的信息，这里，gl就是实体。

在XML中，实体可以分为内部实体、外部实体和参数实体三种类型。内部实体的一般形式为：

```
<!ENTITY  entityName  "will be replaced string">
```

如果被替换的文本很长，可能要把被替换的信息存储在一个文件中。可以通过外部实体参考来实现，即在实体名和文件的URL中使用关键字SYSTEM，构成外部实体，一般形式为：

```
<!ENTITY  entityName  SYSTEM "URL">
```

除此之外，在XML中，还提供了参数实体，它在实体定义中通过在实体名前插入百分号(%)实现，百分号表示该实体为参数实体。一旦被定义，参数定义可以通过用百分号和分号包围参数名来实现。例如：<!ENTITY  % role"(boss | manager | employee)">。

#### 4. 字符数据段

通过预定义XML实体可以在XML文档中加入特殊符号，如果需要大量的特殊符号，

可以使用字符数据段，称为 CDATA 段。CDATA 段可以使用户在一个 XML 文档中引用大量的特殊符号文本块，而不需要分别以实体的形式来代表每一个特殊字符。一般形式为：

```
<![CDATA[text]]>
```

其中，text 是包含特殊字符的文本串，该文本不被 XML 分析器检查。XML 处理器负责分析或者以一种有意义的方式使用该文本块。其中的左右方括号[ ]不能省略。

### 5. 声明并保存外部 DTD 文件

如果希望将 DTD 声明应用到其他 XML 文件中，应该将 DTD 文本保存为一个独立的 .dtd 文件，这样一个 DTD 就可以被多个文档引用，保证版本的一致。对于 .dtd 文件，除了没有内部 DTD 中的<! DOCTYPE rootElement[...]>语句外，其他和一个 XML 文件相同。而且有关元素数目、排列顺序、空元素设定、选择性元素、属性设定、Entity 声明等都和内部 DTD 相同。

当创建了 DTD 文件后，在一个 XML 文档中，可以引用外部 DTD 中声明的元素。要应用一个外部 DTD 文件，需要在 XML 文档的序言中，在 XML 文档内部 DTD 定义和文档内容之前，添加下列外部 DTD 文件声明语句：

```
<!DOCTYPE rootElementName SYSTEM "dtdFile">
```

其中，rootElementName 为应用 DTD 文件的 XML 文档根元素，dtdFile 为要引用的外部 DTD 文件名(.dtd)。

**【例 3-9】** 文档类型定义 DTD 举例。

下面是描述一篇学术论文的 XML 文档，定义了一个 paper 元素及相关属性。代码如下：

```
<?xml version = "1.0" encoding = "gb2312" ?>
<!DOCTYPE paper[
<!ELEMENT paper (TITLE,AUTHOR + ,ABSTRACT?,SUMMARY * )>
<!ATTLIST paper
    paperID CDATA #REQUIRED
    paperStatus (reviewing|accepted) "accepted"
>
<!ELEMENT TITLE (#PCDATA)>
<!ELEMENT AUTHOR (#PCDATA)>
<!ELEMENT ABSTRACT (#PCDATA)>
<!ELEMENT SUMMARY (#PCDATA)>
<!ENTITY KP "knowledge points">
]>

<paper paperID = "p177">
<TITLE>The Research on a Kind of Knowledge Network for Self - learning</TITLE>
<AUTHOR> XW Hao </AUTHOR>
<ABSTRACT>The self - learning is completely student - centered…</ABSTRACT>
<SUMMARY>The idea of self - learning is based on knowledge network mainly relies on manual
construction of &KP;…
</SUMMARY>
</paper>
```

上述代码产生一个严格的 paper 结构，根元素 paper 声明为一个容器元素，包含一个 TITLE 元素、至少一个 AUTHOR 元素、一个可选的 ABSTRACT 元素、零个或多个 SUMMARY 元素。在声明了 paper 元素后，紧接着声明 paper 元素的属性，为了便于阅读，属性的声明往往直接跟在元素声明的后面，虽然这不是必需的。

接下来是声明了一组基本元素，即 TITLE、AUTHOR、ABSTRACT 和 SUMMARY 元素。最后定义了一个实体 KP。

在 IE 浏览器中打开上述文档，显示如图 3-9 所示。

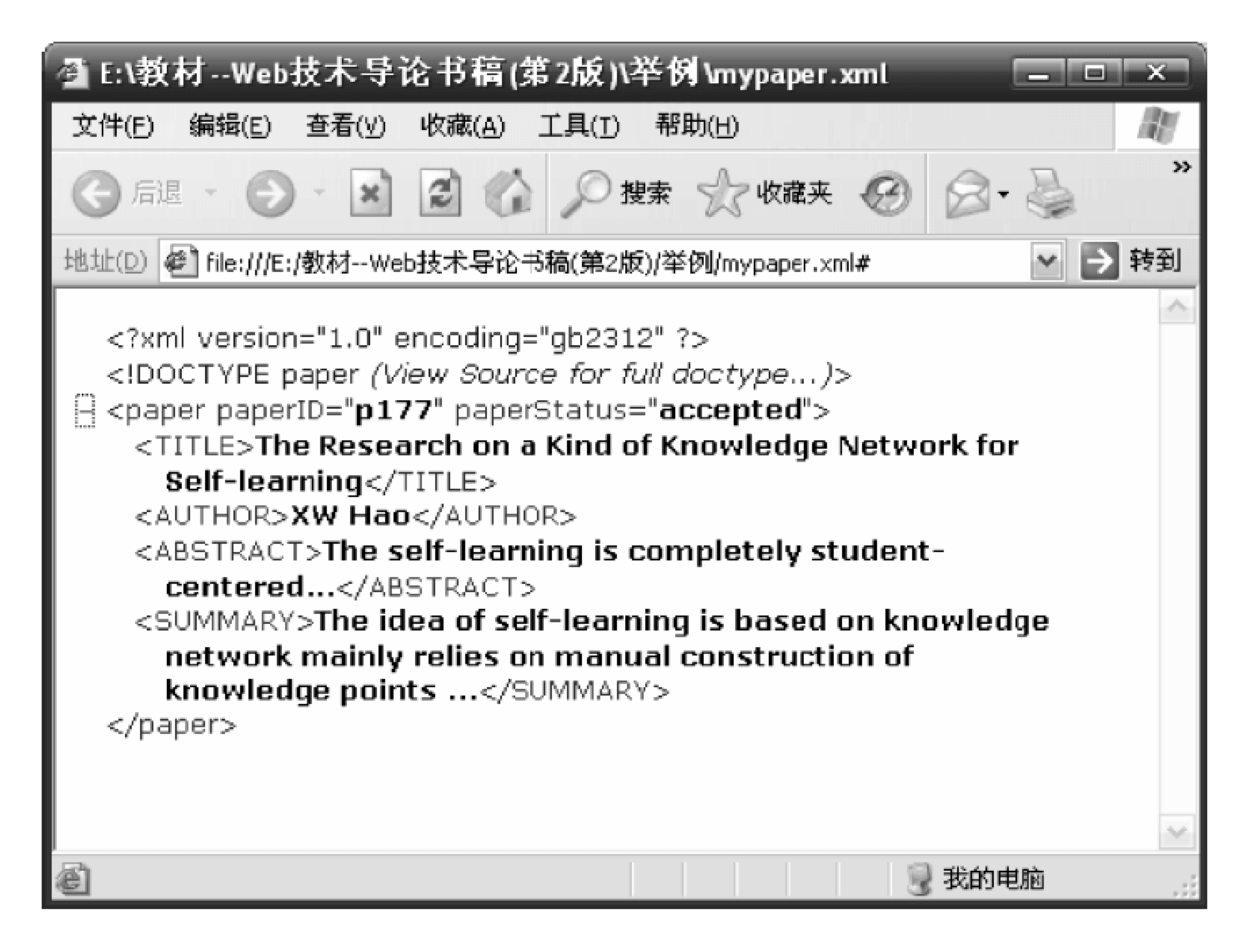



图 3-9　包含 DTD 的 XML 文档

DTD 的语法相当复杂，并且和 XML 文档内容的标记写法不同，自成一个体系，此外，DTD 对元素类型定义不够细致，这些不足导致了 XML 架构的产生。但是，由于现在很多的 XML 应用是建立在 DTD 之上的，能读懂 DTD 文件以及在必要时创建简单的 DTD 文件仍然是很重要的。

### 3.3.4　XML 架构及其应用

在 XML 1.0 规范中，虽然 DTD 实现了 XML 文档中元素(标记)及其类型的定义，对于 XML 文档的结构化起到了很好的描述作用，但是，DTD 支持的数据类型非常有限，扩展性较差，没有一种机制保证数据的应用方和数据的服务方对数据解释的一致。为此，W3C 于 2001 年 5 月正式发布了 XML Schema 的推荐标准，提出了 XML 架构(Schema)的概念，通过 XML Schema 给 XML 数据标注数据类型，从而使得数据的应用方可以通过 XML Schema 规范对所交换的 XML 数据中的信息进行正确的解释，并实现自动处理。利用 XML Schema 规范，Web 服务方和应用方无须事先协调所要交换的数据类型，从而解决了 XML 文档数据的结构和类型一致性解析问题，使 XML 文档结构更加良好。

### 1. XML Schema 的优势

在 XML 中,DTD 和 Schma 都是 XML 文档数据的约束规则,和 DTD 相比,Schema 具有以下优点:(1)XML Schema 对 DTD 进行了扩充,引入了数据类型、命名空间,从而使其具备较强的可扩展性。DTD 提供的数据类型只有 CDATA、Enumerated、NMTOKEN、NMTOKENS 等十种内置(built-in)数据类型。这样少的数据类型通常无法满足文档的可理解性和数据交换的需要。XML Schema 则不同,它提供了更加丰富的数据类型,如 long、int、short、double 等常用的数据类型。(2)XML Schema 利用 Namespace 将文档中特殊的结点与 Schema 说明相联系,一个 XML 文档可以有多个对应的 Schema,而一个 XML 文档只能有一个对应的 DTD。(3)XML Schema 文档本身也是 XML 文档,而不是像 DTD 一样使用特殊格式。开发人员可以使用相同的工具来处理 XML Schema 和其他 XML 信息,而不必专门为 Schema 使用特殊工具。

经过数年的大规模讨论和开发,如今 XML Schema 已经成为全球公认的 XML 环境下首选的数据建模工具。在应用中,有两种主要的 XML Schema,即微软的 XML Schema 和 W3C 的 XML Schema,其中,MS XML Schema 发展较早,支持的软件较多。但是,如果 XML 数据具有很强的开放性,如面向互联网应用,要考虑到对 XML 数据的约束规则今后可以被外部应用所兼容,应尽量选用 W3C 的 XML Schema 规范。

### 2. XML Schema 的一般形式

XML Schema 本身就是一个 XML 文件,不同的是,Schema 文件所描述的是引用它的 XML 文件中的元素和属性的具体类型,即是对于 XML 文档中元素的定义。

如果使用微软的 MS XML Schema,则 XML Schema 定义(XSD)的一般形式为:

```
<?xml version = "1.0"?>
<Schema name = "schema - name"
        xmlns    = "urn:schemas - microsoft - com:xml - data"
        xmlns:dt = "urn:schemas - microsoft - com:datatypes">
  ·
  ·  (XML 元素定义)
  ·
</Schema>
```

定义 Schema 的 XML 文档扩展名为. xml,文档的根元素一定为<Schema>(Schema 的首字母一定大写),用于声明该 XML 文档是一个 Schema 文档。元素<Schema>有三个属性:name 属性给出架构的名称,可省略;xmlns 属性给出了一个微软的架构资源作为一个隐式名称空间,包含了一组预定义元素,包括 Schema、element(声明元素)、ElementType(声明元素数据类型)、AttributeType(声明元素属性数据类型)、attribute、group、datatype、description 等,通过这些元素,定义用户的 XML 元素;xmlns:dt 属性包括微软的一个数据输入资源,作为一个显式名称空间,从而可以引用微软内置的数据类型,包括 string、integer 等。

ElementType 元素用于定义 XML Schema 文档中出现的元素类型,可以是简单元素,也可以是复杂元素。AttributeType 元素用于定义在 Schema 文档中出现的属性类型。

group 元素用于将 XML 文档中的元素分组。datatype 是 XML Schema 中的一个重要元素，也是 XML Schema 的一大特色，它用于为 ElementType 和 AttributeType 指定数据类型。description 的主要作用是为 ElementType 和 AttributeType 元素提供描述信息。

### 3. 定义元素

定义元素就是定义元素的名字和内容模型。在 XML Schema 中，元素的内容模型由其类型定义，定义一个元素，即声明一个元素名称及给定该元素的取值类型，在 XML 文档中实例元素的值必须符合模式中定义的类型。

在 XML Schema 中，定义 XML 元素的一般形式是：

```
<element name='elementname'  type='datatype'/>
```

其中，element 为微软 MS Schema 中的预定义元素，用于声明一个元素，elementname 为要定义的元素的元素名，datatype 声明该元素取值的数据类型。

### 4. 元素类型

元素分为简单类型元素和复杂类型元素两种。简单类型元素的值不能包含元素或属性；复杂类型元素可以产生在其他元素中嵌套元素的效果，或者为元素增加属性。无论是 MS XML Schema 还是 W3C 的 XML Schema 规范，均包含一组预定义的数据类型，除此之外，还允许用户自定义元素数据类型。

(1) 简单类型

简单类型元素是不含属性或其他元素的元素，其数据类型是 XML Schema 规范预定义的数据类型或用户自定义的简单数据类型。XML Schema 规范预定义简单数据类型见表 3-5。

**表 3-5　XML Schema 规范预定义数据类型**

| 数据类型 | 说明 | 数据类型 | 说明 |
|---|---|---|---|
| string | 字符串型，如："hello" | boolean | 布尔类型，如：true、false |
| integer | 整数类型 | date | 日期型，如：2000-02-16 |
| positive-integer | 正整数类型 | time | 时间型，如：15:55:00.000 |
| negative-integer | 负整数类型 | binary | 位模式 |
| non-negative-integer | 非负整数类型，包括 0 | uri-reference | 标准 URL |
| int | 整数类型 | QName | (Prefix ':')? |
| unsigned-int | 无符号整数 | NCName | (letter \| '_') |
| short | 短整数 | ID | 兼容 XML 1.0 DTD |
| unsigned-short | 无符号短整数 | IDREF | 兼容 XML 1.0 DTD |
| byte | 字节类型 | IDREFS | 兼容 XML 1.0 DTD |
| unsigned-byte | 无符号字节类型 | NOTATION | 兼容 XML 1.0 DTD |
| long | 长整数 | ENTITY | 兼容 XML 1.0 DTD |
| unsigned-long | 无符号长整数 | ENTITIES | 兼容 XML 1.0 DTD |
| float | 单精度浮点数 | NMTOKEN | 兼容 XML 1.0 DTD |
| double | 双精度 64 位浮点数 | NMTOKENS | 兼容 XML 1.0 DTD |

除了XMLSchema预定义的简单类型外,用户也可以自定义简单类型。

例如,一般形式是:

```
<ElementType name = "ElementTypeName"
       content = "empty | eltOnly | textOnly | mixed"
       dt:type = "dataType"
       model = "open | closed"
       order = "one | seq | many"/>
```

其中,ElementType为微软MS Schema中的预定义元素,用于声明一个元素类型,ElementTypeName为要定义的元素类型名称,其他各属性说明如下:

① content: 用来描述元素的内容模型。取值可以是:

- empty,元素内容为空;
- textOnly,元素只包含文本内容;
- eltOnly,元素只包含子元素内容;
- mixed,元素既可以包含文本,也可以包含子元素。

② dt:type: 指定元素文本的数据类型,常用的几种数据类型及其含义见表3-5。

③ model: 元素的内容是否遵守Schema中的定义。取值可以是:

- open,元素内容中可以添加未定义过的元素、文本等;
- closed,元素内容中不能添加定义过的元素、文本等。

④ order: 定义子元素的排列顺序。取值可以是:

- one,规定元素内容按多种方式中的某一种排列;
- seq,规定元素内容按指定的顺序排序;
- many,按任意方式排列。

通过简单元素类型,可以声明简单元素,例如:

```
<element name = 'age'  type = 'integer'/>
<element name = 'price' type = 'decimal'/>
```

分别声明两个简单元素<age>和<price>,数据类型分别为integer和decimal。

(2) 复杂类型

在XML Schema中,一个元素如果包含属性或别的元素,这样的元素为复杂类型元素。复杂类型元素又称为容器元素,声明一个复杂类型元素需要定义复杂类型。在XML Schema中,复杂类型的定义和元素声明形式相同。定义一个复杂类型的一般形式为:

```
<ElementType name = "ElementTypeName"
             content = "empty | eltOnly | textOnly | mixed"
             dt:type = "dataType"
             model = "open | closed"
             order = "one | seq | many">
    <attribute name = "AttributeName 1" type = "AttributeTypeName 1">
      ...
    <attribute name = "AttributeName m" type = "AttributeTypeName m">
    <element name = " containedElementName 1" type = "TypeName1">
      ...
    <element name = " containedElementName n" type = "TypeNamen">
</ElementType>
```

其中，ElementType 为微软 MS Schema 中的预定义元素，用于声明一个复杂类型。ElementTypeName 为要定义的复杂类型名称，$AttributeTypeName_{1\dots m}$ 为该复杂类型包含的属性，element 元素声明复杂类型包含的元素，元素类型分别是 $containedElementName_{1\dots n}$。

介绍完了 XML Schema 中声明元素的方法后，下面我们通过一个例子来比较一下 DTD 和 XML Schema 的不同。假定有一个简单的 XML 文档内容如下：

```
<书本>
    <书名>丁丁历险记</书名>
    <作者>Georges Remi</作者>
</书本>
```

如果用 DTD 的形式来定义该 XML 文档结构的话，DTD 定义如下：

```
<!ELEMENT 书本 (书名,作者)>
<!ELEMENT 书名 (#PCDATA)>
<!ELEMENT 作者 (#PCDATA)>
```

如果用 XML Schema 形式来定义 XML 文档结构，则 XML Schema 定义如下：

```
<element name="书本" type="书本类型"/>
<ElementType name="书本类型">
    <element name="书名" type="string"/>
    <element name="作者" type="string"/>
</elementType>
```

其中，元素<书本>的取值类型为用户定义的一个复杂数据类型，因此，<书本>元素为复杂类型元素。

**5. 定义属性类型**

用户可以用属性来描述 XML Schema 元素，在声明属性以前，还需要声明一个属性类型。声明属性类型的语法为：

```
<AttributeType name="attrtypename"
               dt:type="primitive-type"
               dt:values="enumerated-values"
               default="default-value"
               required="{yes|no}">
```

其中，attrtypename 为属性类型名，用于内部引用。primitive-type 是指属性的数据类型，包括：ENTITY、ENTITIES、ENUMERATION（枚举类型）、ID（唯一标识符）、IDREF（标识符引用）、IDREFS（标识符引用集合）、NMTOKEN（名称记号）、NMTOKENS（名称记号集合）、NOTATION、STRING（字符）等。enmuerated-value 是指当属性设为枚举类型时，代表枚举值。default-value 是属性的默认值。required 这个属性标志在运行状态下是否必须存在。

**6. 定义属性**

有了属性类型，就可以在元素中声明属性了，一般形式是：

```
<attribute name = "attrname" type = "attrtypename">
```

其中，attribute 为 Schema 默认元素，用于声明一个元素，有元素名和元素类型两个属性。举例说明如下：

```
<AttributeType name = "location" type = " dt:string"/>
<ElementType name = "communication" content = "textOnly" model = "closed">
    <attribute name = "myaddress" type = "location"/>
</ElementType>
```

上述代码定义了一种元素类型"communication"，该类型不包含其他元素，只包含一个 location 属性，该属性的取值为字符串类型，为复杂元素类型。

**【例 3-10】** 一个 XML Schema 文档(文档名：film. xml)示例。

```
<?xml version = "1.0" encoding = "gb2312" ?>
<Schema xmlns = "urn:schemas - microsoft - com:xml - data"
        xmlns:dt = "urn:schemas - microsoft - com:datatypes">
<ElementType name = "电影" content = "eltOnly" model = "closed" order = "seq">
     <element name = "电影名称" type = "dt:string"/>
     <element name = "导演" type = "dt:string"/>
     <element name = "主演" type = "dt:string"/>
     <element name = "制片公司" type = "dt:string"/>
     <attribute name = "上映日期" type = "dt:date"/>
</ElementType>
</Schema>
```

在上面的例子中，定义了复杂元素类型"电影"，用户还可以定义"近期新片"元素类型，该元素类型可能包含多部电影，对近期影片元素类型的建模请参考 XML 架构的定义。XML Schema 本身是一个 XML 文档，同样可以在浏览器打开，但 XML Schema 的真正目的是系统建模，以及验证 XML 实例文档的数据合法性，参见后面的介绍。

### 7. 使用注释

在 XML 中，注释语句的一般形式是：

```
<!--
  注释文字
  -->
```

### 8. 名称空间

在 XML 中，用户通过 DTD 或者 Schema 自己定义标记和命名元素，这些用户自定义的元素将在别的 XML 文档中被引用。一个 XML 文档可以引用多个 Schema 中定义的元素，不同的 Schema 中定义的元素可能重名，名称空间(Namespaces)就是为此而设计的。

为了避免元素重名，可以将一个 Schema 定义为一个名称空间。名称空间的名字命名必须符合 URL 语法。定义名称空间的唯一目的是唯一地标识一个元素或一组元素的属性，在一个 XML 文档中引用名称空间，需要在 XML 文档的根元素中增加 xmlns 声明，一般形式是：

```
xmlns = "globalUniqueURI"
```

或者

```
xmlns: namespace = "globalUniqueURI"
```

第一种形式的 xmlns 声明没有名字，用于定义该 XML 文档的默认名称空间；第二种 xmlns 声明给定要引用的名称空间一个名字，引用该名称空间的元素和属性，需要在前面增加名称空间的前缀“namespace：”。

例如：

```
<pr: payment xmlns:pr = "http://www.microsoft.com/payroll">
<pr: employee>Jane</pr:employee>
<pr:salary>6000.00</pr:salary>
</pr:payment>
```

有了名称空间，用户就可以保证在文件中使用的名称是唯一的。对元素的属性 xmlns 进行定义就表示对该元素指定了一个名称空间。

### 9. 将架构添加到 XML 文档

用户可以在一个 XML 文档的内部应用一个架构，以便于在运行状态下，XML 验证分析器将架构规则应用于 XML 实例文档，从而验证该文档数据的合法性。语法为：

```
<rootelement xmlns = "x - schema:URI">
```

其中，rootelement 为用户希望应用架构的 XML 文档的根元素。前缀 x-schema 对于微软的 MSXML 处理器是必需的。URI 为统一资源标识符，是要附加的架构的名称。

默认的 xml 名称空间为 xmlns＝"x-schema：URI"，它告诉解析器应该根据 URI 上的 schema(x-schema)来解析整个文档。

**【例 3-11】** 将例 3-10 中的 XML 架构文档(文档名 film. xml)应用到一个 XML 实例文档中(文档名 newfilms. xml)。

```
<?xml version = "1.0" encoding = "gb2312" ?>
<近期新片 xmlns = "x - schema: film.xml">
    <电影 上映日期 = "12 - 20 - 1985">
        <电影名称>警察故事</电影名称>
        <导演>成龙</导演>
        <主演>成龙，张曼玉</主演>
        <制片公司>威禾电影有限公司</制片公司>
    </电影>
    <电影 上映日期 = "12 - 11 - 1992">
        <电影名称>大红灯笼高高挂</电影名称>
        <导演>张艺谋</导演>
        <主演>巩俐</主演>
        <制片公司>西安电影制片厂</制片公司>
    </电影>
</近期新片>
```

在 IE 6 中，newfilm. xml 文档显示如图 3-10 所示。

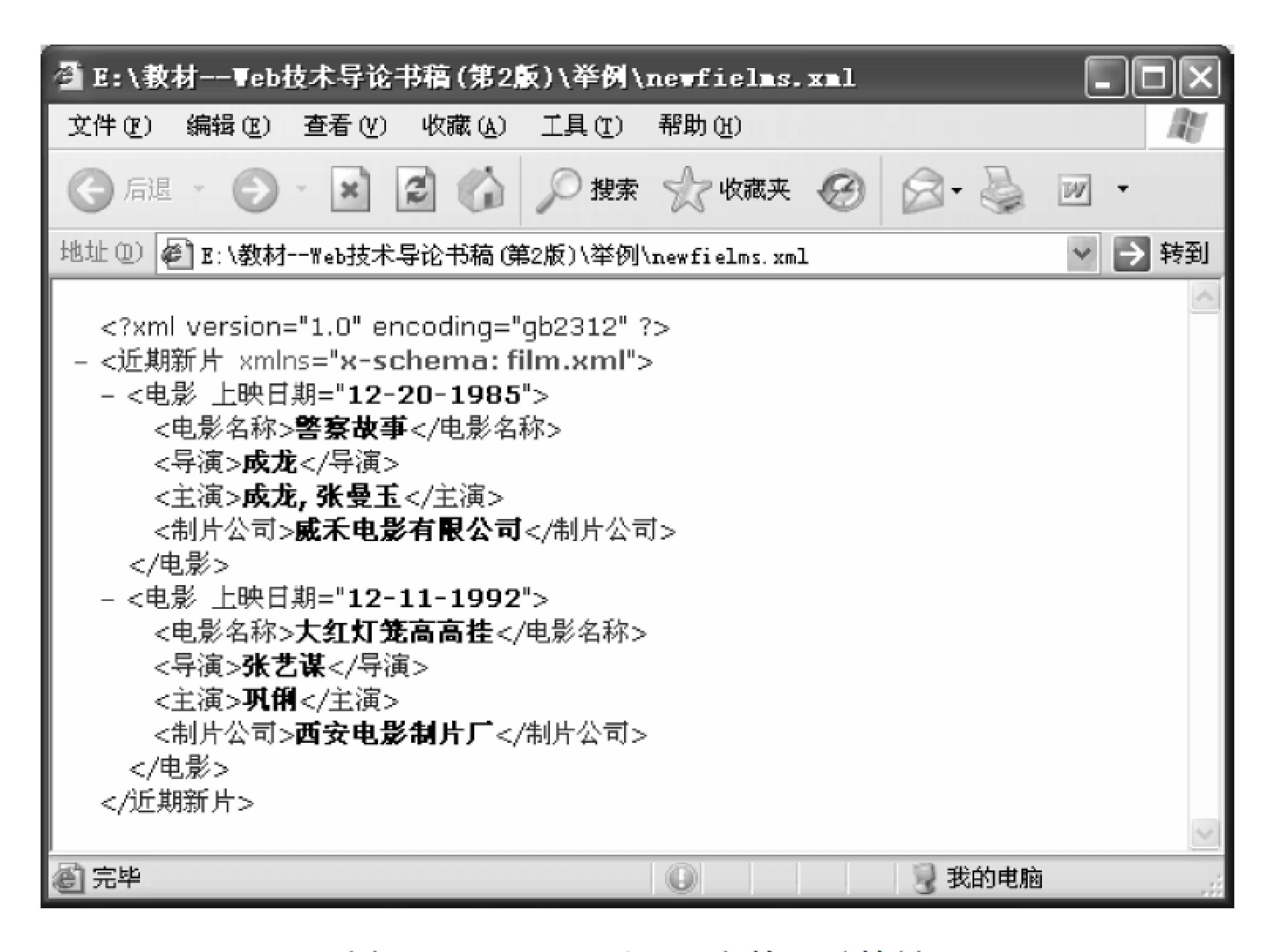



图 3-10 XML Schema 文档显示结果

需要说明的是，浏览器只检查 XML 文档的格式是否正确，不检查 XML 实例文档的有效性。要检查上述 XML 文档是否有效，即是否符合引用的 XML 架构中定义的元素及其类型，可使用 XMLSpy，详细介绍参见 3.4 节。

### 3.3.5 可扩展样式语言 XSL

可扩展样式语言 XSL(eXtensible Style Language)是 W3C 制定的用于描述 XML 文档样式的语言。XSL 语言规范包括扩展样式表转换语言 XSLT(XSL Transformations)和扩展样式表语言格式化对象 XSL-FO(XSL Formatting Objects)两大部分。

#### 1. XSL 的产生

在 Internet 中，XML 已经成为不同应用之间的数据交换标准，从根本上解决了应用系统间的信息交换，保证了数据的平台无关性。为了使数据适合不同的应用程序，我们必须能够将一种数据格式转换为另一种数据格式，比如，需要的格式可能是一个文本文件、一个 SQL 语句、一个 HTTP 信息、一定顺序的数据调用等。

此外，对于 XML 文档数据，我们可能还需要将 XML 数据以不同的方式进行显示。在 XML 技术中，我们可以和 HTML 一样，定义样式表. css 文件，在 XML 文档的序言中增加声明显示该 XML 文档要使用的样式表，语句一般形式为：＜? xml-stylesheet type="text/css" href="mycssfile. css"?＞，然后通过在 XML 元素中使用 class 属性和 ID 属性来定制元素的显示。但是，必须要记住的是：XML 技术的核心是将数据和数据的显示分离，任何在 XML 文档内容中包含显示信息的 XML 文档都认为是含有 bug 的。XML 技术的基本规则是，XML 文档存储数据本身，对数据的显示则通过 XML 数据的应用程序来完成。

基于上述两个方面的原因，W3C 颁布了可扩展样式语言 XSL 规范，用于 XML 文档的转换和显示。XSL 在转换 XML 文档时分为两个过程：首先转换文档结构，然后将文档格式化输出。这两步可以分离开来并单独处理，因此 XSL 在发展过程中逐渐分为 XSLT(结构转换)和 XSL-FO(格式化输出)两种分支语言。XSLT 用来实现 XML 文档结构的转换，其中，将 XML 文档转换为 HTML 文档是目前 XSLT 最主要的功能。XSL-FO 的作用则类似于 CSS 在 HTML 中的作用。

#### 2. XSLT 样式文档基本结构

XSL 样式文档用于处理 XML 文档内容的格式化显示，可以说 XML+XSL 就可以达到 HTML 的显示效果，但 XSL 对数据的显示更灵活，对于同一个 XML 文档，可以编写不同的 XSL，来得到不同的显示。

XSL 样式文档基本结构如下：

(1) 以下面的指令为文档开头(其中还可以包含其他属性)：

```
<?xml version = "1.0"?>
```

(2) 通过 XSLT 元素"xsl：stylesheet"导入 XSL 文档的所有内容，语法为：

```
<xsl:stylesheet xmlns:xsl = "http://www.w3.org/TR/WD-xsl">
```

其中，xmlns：xsl 指明了本 XSL 所采用的标准，在接下来的文档中，通过"xsl："前缀映射到 XSLT 名称空间，使用 XSLT 规范预定义的 XSLT 元素。

(3) 通过一系列模板定义来描述 XML 文档的显示格式。在模板中包含了大量的 HTML 标记，通过 xsl：for-each、xsl：if、xsl：choose 等 XSLT 元素进行数据的循环处理、条件处理、选择处理等工作，定位 XML 文档中的内容。

(4) 通过调用相应的模板，完成 XML 文档内容的输出。

XSL 样式文档文件扩展名为.xsl，要使用 xsl 显示一个 XML 文档，在 XML 文档声明的后面需要增加一个声明语句，声明用于显示该 XML 文档的 xsl 程序，一般形式如下：

```
<?xml-stylesheet type = "text/xsl" href = "xslfile.xsl"?>
```

其中，xslfile.xsl 是用于该 XML 文档输出的 XSLT 文件。

#### 3. XSLT 元素

在 XSLT 规范中，定义了大量的 XSLT 元素和 XSLT 函数，用于对 XML 文档的转化，下面我们对其中常用的一些元素进行简要介绍。

(1) <xsl：stylesheet>元素

指定 XSLT 文件的文档元素，该文件中包含所有其他 XSLT 元素。一般形式是：

```
<xsl:stylesheet id = "id" extension - element - prefixes = "NCNames" >
…
</xsl:stylesheet>
```

其中的属性意义如下：

- id：嵌入 XSLT 文件标识符。

- extension-element-prefixes：要作为扩展命名空间使用的命名空间。

(2) <xsl：template>元素

定义一个可再次使用的模板，用于为特定类型和上下文的结点生成所需的输出。XSLT 文件就是由一个一个的模板组成的，任何一个 XSLT 文件至少包含一个模板。模板由匹配模式(match pattern)和执行两部分组成，模式定义 XML 源文档中哪一个结点将被模板处理，执行则定义输出的是什么格式。

模板定义的一般形式为：

```
<xsl:template  name = "Qname" match = "Pattern" priority = "number" mode = "Qname">
</xsl:template>
```

其中的属性意义如下：

- name：模板名称，为可选属性。如果使用<xsl：call-template>元素调用模板，则需要指定要调用的模板名称，因此在定义模板时，需要定义模板的名称。如果使用<xsl：apply-templates>元素调用模板，则不需要模板名称。
- match：确定什么样的情况下执行此模板。作为一种简化的说明，在此处通常使用元素的名字；其中最上层模板必须将 match 设为“/”。在一个 XSL 文档中根模板是唯一的。match 属性是必选项，除非 <xsl：template>元素有 name 属性。

<xsl：template>用 match 属性从 XML 选取满足条件的结点，针对这些特定的结点形成一个特定输出形式的模板。在一个 XSLT 文档中，一般要设计多个模板，在各个模板结构中，描述了不同层次的元素的数据的显示格式、数据引用、数据处理等内容。

- priority：模板的优先级编号，此属性值为 0～9 的实数。
- mode：模式值，允许多次处理某个元素，每次产生不同的结果。如果<xsl：template>没有 match 属性，就不得有 mode 属性。如果<xsl：apply-templates>元素有 mode 属性，则只应用于 mode 属性值相同的<xsl：template>元素中的模板规则；如果<xsl：apply-templates>元素没有 mode 属性，则只应用于没有 mode 属性的<xsl：template>元素中的模板规则。

(3) <xsl：apply-templates>元素

指示 XSLT 处理器根据每个选定结点的类型和上下文，从<xsl：template>元素定义的一系列模板中找到适合应用的模板，一般形式为：

```
<xsl:apply - templates select = "expression" mode = "Qname">
```

其中的属性意义如下：

- select：确定应调用的模板，即选取用<xsl：template>标记建立的模板。可以用于处理通过表达式选择的结点，而不是处理所有子级。select 属性的值是表达式，该表达式必须计算为结点集。除非指定了不同的排序顺序，否则，选定的结点集按文档顺序进行处理。
- mode：该属性允许多次处理其限定名指定的某个元素，每次产生不同的结果。如果<xsl：template>没有 match 属性，就不得有 mode 属性。如果<xsl：apply-templates>元素有 mode 属性，将只应用于 mode 属性值相同的<xsl：template>元素中的模板规则。如果 <xsl：apply-templates>元素没有 mode 属性，将只应用

于没有 mode 属性的<xsl：template>元素中的模板规则。

<xsl：apply-templates>元素先使用 select 属性中指定的表达式选择一组结点。如果未指定此属性，将选择当前结点的所有子级。<xsl：apply-templates> 指示 XSLT 处理器为每个选定的结点找到适合应用的<xsl：template>。通过将结点与模板的 match 属性中指定的 XPath 表达式进行比较，测试模板是否适用。如果多个模板满足匹配模式，将选择其中优先级最高的模板。如果多个模板的优先级相同，将选择样式表中的最后一个模板。

应用模板<xsl：apply-templates>可以将其理解为程序中的函数调用。如果模板定义了模板名，也可以通过<xsl：call-template>元素调用一个特定名称的模板。

(4) <xsl：include>元素

在一个 XSLT 文件中包含另一个 XSLT 文件。一般形式是：

```
<xsl:include href = "uri - reference"/>
```

Href 属性为必选项，标识要包含的 XSLT 文件的统一资源标识符(URI)引用。

(5) 选择模式相关元素

选择模式就是用选择的方式将数据从 XML 中提取出来，这是一种在 XSL 中应用广泛且操作简单的获得数据的方法。选择模式有三种不同的元素，各自的语法格式如下：

① <xsl：value-of>元素

从 XML 文档中提取指定结点的数据，一般形式是：

```
<xsl:value - of select = "Expression">
```

Select 属性用来指定提取的元素，从而将源文档中元素的文本值写到输出文档中。如果表达式返回多个结点，<xsl：value-of> 元素将返回所返回的第一个结点的文本。如果返回的结点是包含子结构的元素，<xsl：value-of> 返回该元素子树的串联文本结点，并删除标记。

② <xsl：for-each>元素

循环处理被选择的元素结点，一般形式是：

```
<xsl:for - each select = "Expression">
```

其中，select 属性为必选项，表达式在当前上下文中计算，确定要循环访问的结点集。

③ <xsl：apply-templates>元素

调用<xsl：template>元素定义的模板，用来指定哪一个结点被模板具体处理。可以将它理解为程序中调用子函数。一般格式如下：

```
<xsl:apply - templates select = "Expression" >
```

select 属性通常为结点集，用来定义要应用模式的结点名称。

(6) 测试模式相关元素

测试模式有以下两种元素：

① <xsl：if>元素

测试 test 属性给定的条件，一般形式是：

```
<xsl:if test = " boolean - expression ">
```

当条件的值为 True 时执行该元素的内容，否则不执行该元素的内容。如果存在 script 属性，则可以以一句脚本语言表达式为测试条件。默认使用的脚本语言为 JScript。

对于测试条件，情况比较复杂，常用的为一个关系表达式或逻辑表达式，其书写规则如下面的例子所示：

(a) 姓名[="张三"](元素"姓名"的值为"张三")；

(b) 成绩[. $ge$ 90](元素"成绩"的值大于或等于 90)；

(c) 成绩[. $gt$ 90 or. $lt$ 60](元素"成绩"的值大于 90 或小于 60)；

(d) @性别[. ="女"](当前元素的"性别"属性值为"女"时为真)；

XSLT 中的运算符较多，包括：选择运算符和特殊字符、逻辑运算符、关系运算符以及集合运算符等，详细信息请参考 XSLT 规范。

② <xsl: choose>元素

和 xsl: when 元素一起，用于多个条件的测试，执行满足条件的元素内容，一般形式是：

```
<xsl:choose>
        <xsl:when test = "boolean-expression">
        <xsl:othrwise>
```

将按照从上到下的顺序测试<xsl: choose>元素的<xsl: when>子级，直到其中一个元素上的 test 属性准确地说明源数据中的条件，或直到遇到<xsl: otherwise>元素。在选择了<xsl: when> 或<xsl: otherwise>元素之后，将退出<xsl: choose>块。不需要明确地中断或退出语句。

(7) <xsl: element>元素

添加一个 HTML 元素，一般形式是：

```
<xsl:element name = "element - name" namespace = "uri - reference" use - attribute - sets = QName
</xsl:element>
```

各属性含义如下：

- name：必选项。要创建的元素的名称。
- namespace：所创建元素的命名空间 URI。如果 name 属性包含 QName，指定的前缀将绑定到 namespace 属性所指定的命名空间上。
- use-attribute-sets：通过空白分隔的属性集列表。

(8) <xsl: attribute>元素

声明一个元素属性，一般形式是：

```
<xsl:attribute name = "attribute - name" namespace = "uri - reference">
</xsl:attribute>
```

其中，name 必选项为要创建的属性的名称，namespace 为已创建属性的命名空间统一资源标识符(URI)。

**【例 3-12】** 模板相关元素举例。

有一 XML 文档 mybooks. xml 代码如下：

```
<?xml version = "1.0"?>
```

```
<?xml-stylesheet type="text/xsl" href="xslinclude.xsl"?>
<COLLECTION>
   <BOOK>
      <TITLE>The Adventures of Tom Sawyer </TITLE>
      <AUTHOR>Twain,Mark </AUTHOR>
   </BOOK>
   <BOOK>
      <TITLE>The Little Match-Seller </TITLE>
      <AUTHOR>Andersen,Hans C.</AUTHOR>
   </BOOK>
</COLLECTION>
```

XSLT 文件 xslinclude.xsl 代码如下：

```
<?xml version='1.0'?>
<xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="xml" omit-xml-declaration="yes"/>
<xsl:template match="/">
    <xsl:for-each select="COLLECTION/BOOK">
        <xsl:apply-templates select="TITLE"/>
        <xsl:apply-templates select="AUTHOR"/>
        <br/>
    </xsl:for-each>
</xsl:template>
<!--The following template rule will not be called,because the related template
in the including stylesheet will be called.If we move this template so that
   it follows the xsl:include instruction,this one will be called instead.-->
<xsl:template match="TITLE">
   <div style="color:blue">
      Title: <xsl:value-of select="."/>
   </div>
</xsl:template>
<xsl:include href="xslincludefile.xsl" />
</xsl:stylesheet>
```

包含的 XSLT 文件(xslincludefile.xsl)代码如下：

```
<?xml version='1.0'?>
<xsl:stylesheet version="1.0"
         xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
         xsl:space="preserve">
<xsl:template match="TITLE">
    Title-<xsl:value-of select="."/><br/>
</xsl:template>
<xsl:template match="AUTHOR">
    Author-<xsl:value-of select="."/><br/>
</xsl:template>
</xsl:stylesheet>
```

输出结果如图 3-11 所示。

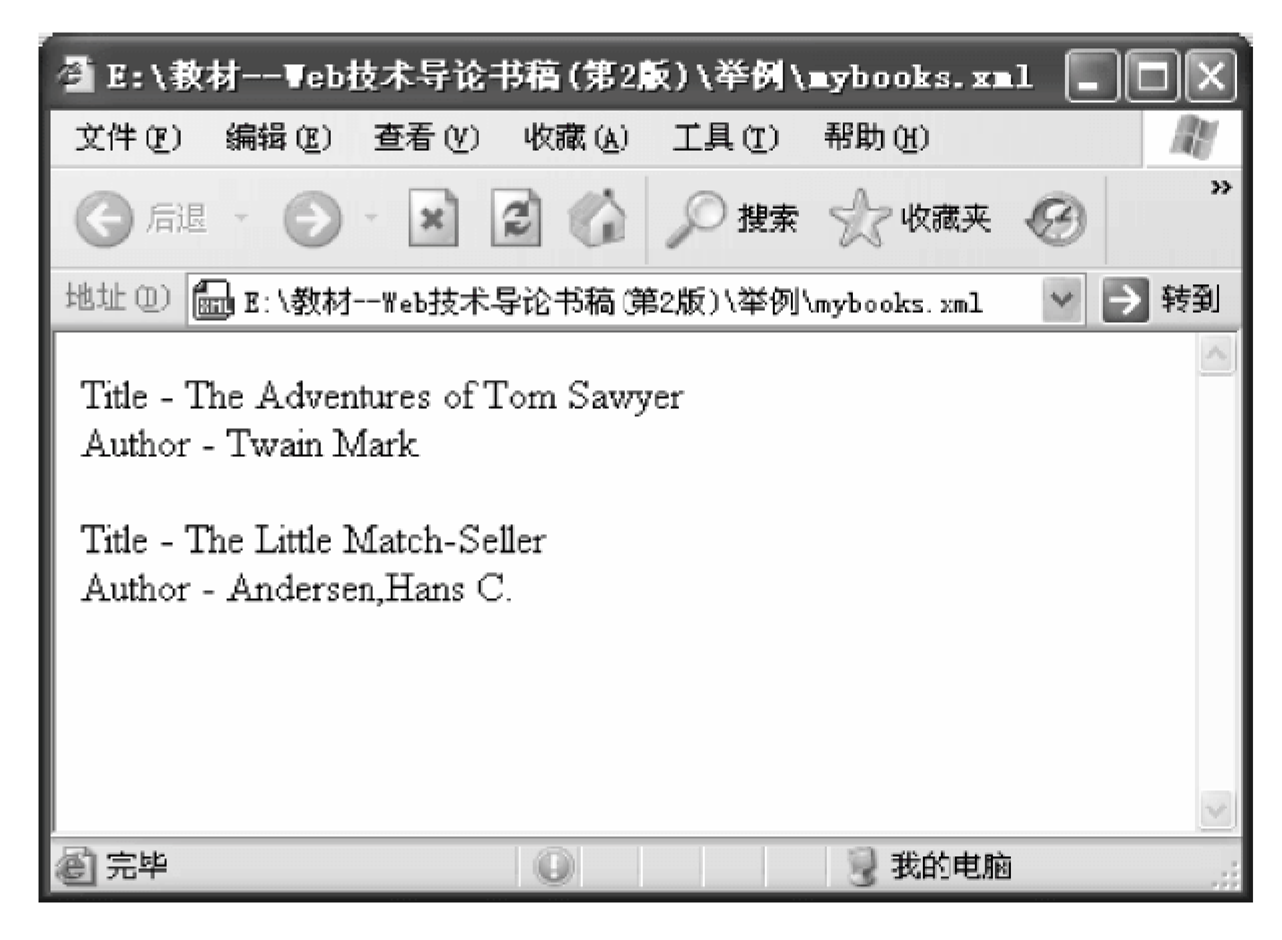



图 3-11　XML 文档的 XSLT 输出界面

**【例 3-13】** XSLT 综合举例。

下面是一个描述网络课程知识单元的 XML 文档,通过 XSLT 设计其显示界面。

**清单 1:** 知识单元 XML 文档内容清单(KUxxyyzz.xml)

```
<?xml version = "1.0" encoding = "gb2312" ?>
<?xml - stylesheet type = "text/xsl" href = "KUpage.xsl" ?>
<KU>
<title>2.1 The OSI model</title>
<LearningObjectives>
   <objectiveitem><goal>(1) 了解 OSI 七层模型</goal></objectiveitem>
   <objectiveitem><goal>(2) 掌握各层名称及其功能</goal></objectiveitem>
   <objectiveitem><goal>(3) 简单了解数据封装过程</goal></objectiveitem>
</LearningObjectives>
<content>
```

20 世纪 80 年代末 90 年代初,<kp>计算机网络</kp>迅猛发展。为了帮助网络厂商建构网络以及支持网络互操作,<kp>ISO 组织</kp>定义了异质系统互连的七层框架体系结构标准,也称为<kp>OSI 参考模型(OSI Reference Model)</kp>。

```
    …
</content>
<KPList>
<kpitem>
<kp> OSI model </kp>
<los>
<loitem>
<loname>OSI 模型七层划分</loname>
<url>../../lor/计算机网络/L005 - 01 - 005 - 01.jpg</url>
<tar>frameA</tar>
</loitem>
<loitem>
<loname>OSI 模型各层功能</loname>
<url>../../lor/计算机网络/L005 - 01 - 005 - 02.swf</url>
<tar>frameA</tar>
</loitem>
```

```
<loitem>
<loname>OSI 模型各层设备</loname>
<url>../../lor/计算机网络/L005 - 01 - 005 - 03.swf</url>
<tar>frameA</tar>
</loitem>
</los>
</kpitem>
<kpitem>
<kp>Encapsulation</kp>
<los>
<loitem>
<loname>数据封装</loname>
<url>../../lor/计算机网络/L005 - 01 - 006 - 01.ppt </url>
<tar>frameA</tar>
</loitem>
</los>
</kpitem>
</KPList>
<FigName>图 2 - 1 互联网概念模型</FigName>
<ImgPath>fig2 - 1.jpg</ImgPath>
</KU>
```

下面是用于显示上述 XML 文档的 XSLT 程序清单。

**清单 2：** XSLT 文档 KUpage.xsl 内容清单

```
<?xml version = "1.0" encoding = "GB2312"?>
<xsl:stylesheet xmlns:xsl = "http://www.w3.org/TR/WD - xsl">
<xsl:template match = "/" language = "javascript">
   <html>
   <head><title>知识单元</title></head>
   <body>
   <table border = "0" width = "100%" cellspacing = "0" cellpadding = "0">
   <tr>
   <td height = "26" bgcolor = "#CC0033">
   <font color = "#FFFFFF"> <xsl:value - of select = "KU/title"/> </font>
   </td>
</tr>
<tr height = "30"><td><b>学习目标</b></td></tr>
 <xsl:for - each select = "KU/LearningObjectives/objectiveitem">
   <tr><td height = "30"><xsl:value - of select = "goal"/></td></tr>
 </xsl:for - each>
</table>
<table width = "100%" id = "table1" style = "border - top - style: solid; border - top - width: 1px;
      border - bottom - style: solid; border - bottom - width: 1px" >
<tr> <td><xsl:value - of select = "KU/content"/></td></tr>
</table>
<table>
<tr height = "30"><td colspan = "2"><b>本单元所包含的知识点</b></td></tr>
<tr> <td>知识点</td> <td>学习对象</td></tr>
<xsl:for - each select = "KU/KPList/kpitem">
  <tr>
  <td><font color = "green"><b><xsl:value - of select = "kp"/></b></font></td>
  <xsl:for - each select = "los/loitem">
  <td>
```

```
        <a>
            <xsl:attribute name = "href"><xsl:value - of selct = "url" />
            </xsl:attribute>
            <xsl:value - of select = "loname"/>
        </a>
    </td>
    </xsl:for - each>
    </tr>
   </xsl:for - each>
  </table>
  <table>
    <tr height = "30"><td colspan = "2"><b>本单元附加文件</b></td></tr>
    <tr><td><xsl:value - of select = "KU/FigName"/></td></tr>
    <tr>
        <td>
            <xsl:element name = "img">
              <xsl:attribute name = "src"><xsl:value - of select = "KU/ImgPath"/></xsl:attribute>
            </xsl:element>
        </td>
      </tr>
     </table>
    </body>
    </html>
  </xsl:template>
  </xsl:stylesheet>
```

当 XML 文档在浏览器中打开时，显示出的页面如图 3-12 所示。

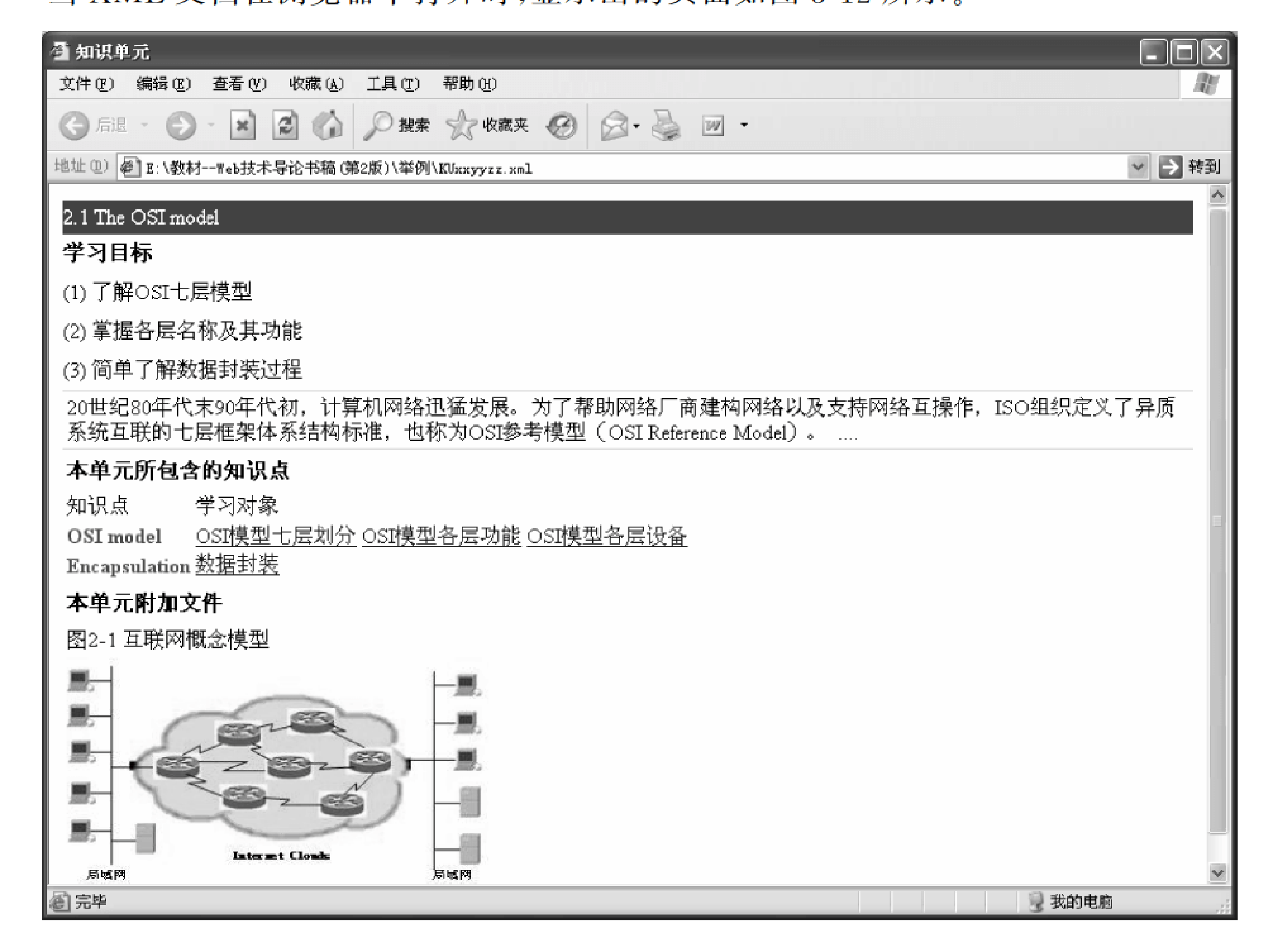



图 3-12　XML 文档使用 XSLT 的输出界面

上述 XML 文档定义了一个知识单元的结构，包括知识点，每一个知识点都对应一个或多个学习对象，每一个学习对象对应一个指向学习对象媒体文件的超链接，超链接<a>属性是通过<xsl: attribute>元素来设定的。知识单元还可以附加本知识单元的一些图、表等，对于 XML 中图片的显示，和超链接的显示不同，例子给出了另外一种方法，即通过<xsl: element name="img">来显示。

## 3.3.6　XML 路径语言 XPath

在 XSLT 和 XPointer 规范中，都需要定位 XML 文档树中的元素、元素属性或元素区间，XPath 规范的目的就是向 XSLT 和 XPointer 共同需要的功能提供统一的语法和语义，可以说，XPath 是一种 XML 文档内容寻址语言，主要应用于 XSLT 和 XPointer 的规范中。XPath 将一个 XML 文档建模成为一棵结点树，用位置路径来实现对 XML 文档内容的定位，从而实现对 XML 文档中的元素和属性进行遍历，实现对 XML 文档内容的寻址。

### 1. XPath 语法

XPath 是 XML 的查询语言，和 SQL 的角色很类似。XPath 使用路径表达式在 XML 文档中选择结点或结点集。下面以一个简单的 XML 文档为例，介绍 XPath 的基本语法。

```
<?xml version="1.0"?>
<bookstore>
<book>
    <title lang="eng">Aladdin and the Enchanted Lamp</title>
    <price>2.90</price>
</book>
<book>
    <title lang="eng">Huckleberry Finn</title>
    <price>2.80</price>
</book>
</bookstore>
```

(1) 基本 XPath 表达式

XPath 表达式是使用运算符和特殊字符构造的，用于确定查询范围，即选择 XML 中的文档结点集合。XPath 运算符和特殊字符见表 3-6。

**表 3-6　XPath 路径运算符和特殊字符**

| 运算符或特殊字符 | 说　明 | 运算符或特殊字符 | 说　明 |
|---|---|---|---|
| / | 子运算符，在左侧集合中的直接子结点元素中，选择运算符右侧的指定元素。当此路径运算符出现在模式开头时，表示选择根结点下的直接子元素 | : | 命名空间分隔符，将命名空间前缀与元素名或属性名分隔 |
| // | 递归下降，在左侧集合的任意深度子结点中搜索运算符右侧的指定元素。当此路径运算符出现在模式开头时，表示应从根结点递归下降 | () | 为运算分组，明确设置优先级 |

续表

| 运算符或特殊字符 | 说　明 | 运算符或特殊字符 | 说　明 |
|---|---|---|---|
| . | 表示当前上下文结点元素自身 | [] | 应用筛选模式,或下标运算符 |
| .. | 当前上下文结点的双亲元素 | +,-,* | 算术运算符 |
| * | 通配符,选择所有元素或属性 | DIV | 除法 |
| @ | 属性名前缀 | MOD | 求模 |

下面是一组 XPath 的常用表达式,说明如下:

- 文档根:以正斜杠(/)为前缀的表达式使用文档树的根作为上下文。需要说明的是,根结点不是根元素,它是一个抽象的结点,是根元素的父结点,用"/"表示。

例如,/bookstore,选择根元素 bookstore;bookstore/book,选择 bookstore 下的所有 book 子元素;/bookstore/*,选择 bookstore 元素下的所有子结点。

- 根元素:使用正斜杠后接星号(/*)的表达式使用根元素作为上下文。

例如,/*,表示查找文档的根元素。

- 特定元素:以元素名开头的表达式引用特定元素的查询,从当前上下文结点开始,选择当前结点的所有子结点。

例如,bookstore,表示选择 bookstore 元素的所有子结点。

- 递归下降:使用双正斜杠(//)的表达式,表示从当前结点开始选择符合条件的所有结点元素。如果此运算符出现在模式的开头,上下文相对于文档的根。

例如,//book,表示选择文档中的所有 book 元素;//*,表示选择文档中的所有元素;bookstore//book,表示选择 bookstore 下的所有 book 子孙元素。

.//前缀则表明上下文从层次结构中当前上下文所指示的级别开始。

- 选择属性://@lang,表示选择所有名为 lang 的属性;//title[@*],表示选择所有 title 元素,不管其属性如何。

在上面的表述中,所谓文档上下文是指定位计算所基于的一个已知结点,该结点称为上下文结点。一般而言,最初始的上下文结点总是文档中的确定位置,如文档的根结点。

(2) 谓词

谓词用于寻找特定的结点或一个包含特定值的结点,谓词包括在方括号中,例如:

/bookstore/book[1],表示选择 bookstore 元素下的第一个 book 子元素。

/bookstore/book[last()],表示选择 bookstore 元素下的最后一个 book 子元素。

/bookstore/book[position()<3],表示选择 bookstore 元素下开始的两个 book 子元素。

//title[@lang],表示选择所有的包含属性名为 lang 的 title 元素。

//title[@lang='eng'],表示选择所有的包含属性名为 lang 且属性值为 eng 的 title 元素。

/bookstore/book[price>2.00],表示选择 bookstore 元素下所有包含值大于 2.00 的 price 元素的 book 子元素。

/bookstore/book[price>2.00]/title,表示选择 bookstore 元素下的所有包含值大于 2.00 的 price 元素的 book 子元素下的 title 子元素。

(3) 多路径选择

如果需要选择多个路径,可以使用"|"操作符,例如:

//book/title|//book/price,表示选择所有 book 元素下的 title 和 price 子元素。

//title|//price,表示选择文档中的所有的 title 元素和 price 元素。

/bookstore/book/title|//price,表示选择/bookstore/book/下的 title 子元素和文档中的所有 price 元素。

(4) XPath 轴心

一个轴心(axis)定义了相对于当前上下文结点的结点集,各轴心名及含义如下:

- ancestor 轴,表示选择当前结点的所有祖先。
- ancestor-or-self 轴,表示选择当前结点及其所有祖先。
- attribute 轴,表示选择当前结点的属性。
- child 轴,表示选择当前结点的所有子女。
- descendant 轴,表示选择当前结点的所有子孙。
- descendant-or-self,表示选择当前结点及其所有子孙。
- following 轴,表示选择上下文结点后面的元素结点。
- following-sibling 轴,表示选择当前上下文结点后面与上下文结点同属于一个父结点的那些结点。
- namespace 轴,表示选择当前结点的所有名字空间结点。
- parent 轴,表示选择当前结点的双亲。
- preceding 轴,表示选择当前结点前面的结点。
- preceding-sibling 轴,表示选择当前结点前面与上下文结点同属于一个父结点的那些结点。
- self 轴,表示选择当前结点,也可以用". "来表示。

XPath 轴心应用于位置路径表达式中,来确定一个位置步中的结点集。

**【例 3-14】** 使用 XPath 举例。

有下面的 XML 文档 bookstore. xml,代码如下:

```
<?xml version = "1.0" encoding = "gb2312"?>
<?xml - stylesheet type = "text/xsl" href = "bookstore.xslt"?>
<bookstore>网上书店
    <名称>World famous novels</名称>
    <图书 日期 = "2003 - 11 - 24">
        <名称>ROMEO AND JULIET </名称>
        <作者>Shakespeare,William </作者>
        <价格>30.00</价格>
    </图书>
    <图书 日期 = "2006 - 5 - 8">
        <名称>The Mysterious Island </名称>
        <作者>Verne,Jules </作者>
        <价格>31.00</价格>
    </图书>
</bookstore>
```

用于输出的 bookstore. xslt 代码如下:

```
<?xml version = "1.0" encoding = "gb2312"?>
<xsl:stylesheet version = "1.0" xmlns:xsl = "http://www.w3.org/1999/XSL/Transform">
    <xsl:template match = "/">
          <html>
                <head>
                   <title>练习使用 XPath 轴</title>
                </head>
                <body>
                     <xsl:apply - templates select = "bookstore"/>
                </body>
          </html>
    </xsl:template>
<xsl:template match = "bookstore">
       <xsl:apply - templates select = "child:: * "/>
</xsl:template>
<xsl:template match = "text()">
       Textnode: <xsl:value - of select = "self::text()"/> <br/>
</xsl:template>
<xsl:template match = " * ">
       Booknode: <xsl:value - of select = "self:: * "/> <br/>
</xsl:template>
</xsl:stylesheet>
```

在 IE 浏览器中的输出结果如图 3-13 所示。

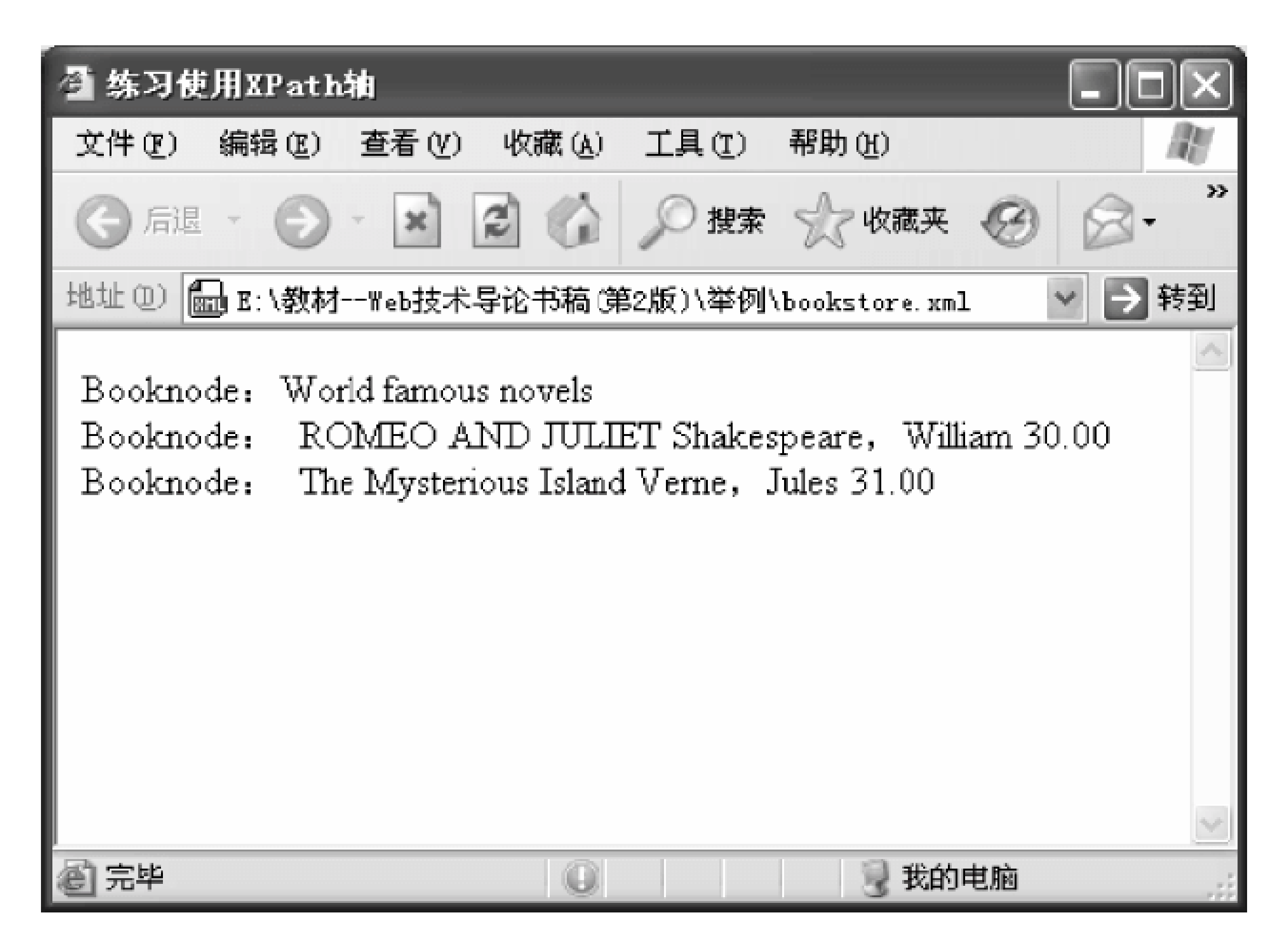



图 3-13 XPath 轴的应用输出结果界面

(5) 位置路径

位置路径(Location Path)是由"/"分隔的位置步(Location Step)构成的一个表达式,可用于 XSL 和 XPointer 表达式中。位置步在目标文件中指定一个位置,通常是相对于一个已知的位置,如文件的根结点或另外一个位置步等。位置步由一个 XPath 轴心关键字(axis)、结点测试(node-test)和可选谓词(prediates)构成,一般形式如下:

```
axis::node - test[prediates]
```

例如，/child∷spec/child∷body[position()=2]

其中，第一个"/"表示根结点；"child∷spec"是第一个位置步，表示根结点的 spec 直接子元素，如果文件的根元素是 sepc，计算出来的结果应为根元素，否则计算结果为空；"child∷body[position()=2]"是第二个位置步，表示上次计算出来元素的第二个 body 直接子元素。在上面的例子中，如果文件的根元素是 spec，而且它包含两个以上的 body 子元素，返回的结果将是一个元素，否则将不返回结果。

## 2. XPath 函数

XPath 定义了一组内置函数，称为核心函数库。在 XPath 1.0 中，包含四类函数，分别是：结点集函数、字符串函数、布尔函数和数字函数。XPath 2.0 对函数进行很大的扩充，分成七类，包含 100 多个函数，这些函数针对字符串值、数字值、日期和时间比较、结点操作、顺序操作、布尔值等等。其中最常用的结点集函数有：

(1) last()：返回上下文大小，即给定上下文中的结点数。

(2) position()：返回上下文位置，即当前结点在给定上下文结点集(列表)中的位置。比如，可以用表达式 position()=last()测试处理的是否是集合中的最后一个结点。

(3) count(node-set)：返回实参结点集中的结点数。

(4) id(object)：返回一个结点集，根据在 DTD 中声明为 ID 类型的唯一标识符选择元素。

关于 XPath 的函数的详细说明和使用请参考 XPath 规范。

**【例 3-15】**　XPath 函数应用举例。

在以下 XML 文档(exaxpath.xml)中选择所有值为 green 或 blue 的<x>元素，并输出。

```
<?xml version = '1.0'?>
<?xml - stylesheet type = "text/xsl" href = "exaxpathstyle.xsl"?>
<root>
    <x>green</x>
    <y>
        <x>blue</x>
        <x>blue</x>
    </y>
    <z>
        <x>red</x>
        <x>red</x>
    </z>
    <x>green</x>
</root>
```

清单：exaxpathstyle.xsl

```
<?xml version = '1.0'?>
<xsl:stylesheet version = "1.0"
        xmlns:xsl = "http://www.w3.org/1999/XSL/Transform">
<xsl:template match = "root">
    <xsl:for - each select = "x | y/x">
```

```
        <xsl:value-of select="."/>,
        <xsl:if test="not(position() = last())">,</xsl:if>
    </xsl:for-each>
</xsl:template>
</xsl:stylesheet>
```

则在浏览器中的格式化输出为：green,,blue,,blue,,green,

### 3.3.7 XML 查询语言 XQuery

现在，越来越多的信息以 XML 格式进行存储和交换，XML 数据查询成为重要的功能需求。1999 年，W3C 成立了 XML 查询工作组，开始研究制定相关标准规范。经过一个漫长的时期，2005 年 11 月，W3C 发布了关于 XML 查询的 8 个备选推荐规范。2007 年 1 月 23 日，W3C 才将 XPath 2.0 和 XQuery 1.0 确定为推荐标准。

XQuery 1.0 是 XPath 2.0 的扩展集，除了拥有 XPath 2.0 的特点外，增加了排序、重装、构造功能，而且实现了 XPath 2.0 未能实现的数据浏览和过滤方面的性能。XQuery 的优点包括：

- 相比 XSLT 的查询语句，XQuery 查询语句代码更简洁。XQuery 执行查询需要的代码比 XSLT 少，所以它的执行效率也高。
- 当 XML 数据是类型化的，那么 XQuery 是一个强类型语言，它能够通过避免非法的类型转换以及确认类型是否可以在查询操作中使用，来提高查询语句的执行效率。
- XQuery 能当作弱类型语言使用，为非类型化数据提供更强的功能。
- XQuery 正在成为 W3C 工作组的推荐语言，同时它也将会被主流的数据库提供商所支持。

由于本书篇幅所限，关于 XQuery 的基本语法和应用请参考 W3C 的具体规范。

### 3.3.8 可扩展连接语言 XLL

在 XML 的规范中，并没有规定有关文件链接的问题。为了使 XML 文件也能够有类似 HTML 文件超链接的功能，W3C 制定了 XML 可扩展链接语言规范 XLL(eXtensible Linking Language)，分为三个部分：XLink 语言、XPointer 语言和 XML Base。其中 XLink 是规定 XML 文件之间的链接规范(和 HTML 中的外链接相似)，XPointer 是规定 XML 文件中不同位置之间的链接规范(类似 HTML 中的书签)。通过 XLink 规范和 XPointer 规范可以定义类似 HTML 中<a>标记的超链接功能，然后通过 XSL 显示该超链接。

#### 1. XML 文档链接 XLink 规范

XLink 所设定的链接分为简单链接(Simple Link)和扩展链接(Extended Link)。其中，Simple Link 的链接功能和 HTML 的超链接基本上一样，将单个源文档和多个目标文档相链接。Extended Link 则超出了 HTML 超链接的功能，它链接的对象可以一次设定多个，允许在多个源文档和多个目标文档之间建立链接。

下面我们主要介绍 XLink 中的简单连接的定义和显示方法。

在 XML 文件中定义使用 XLink 元素的时候，必须要在 DTD 中声明连接元素，下列 XML 代码声明了一个 Simple Link 类型的 XLink 元素<friendlink>。然后，通过<friendlink>元素标记 XML 文档中的超链接。XML 文档名为 myfriends. xml，代码清单如下：

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="friendlists.xsl" ?>
<!DOCTYPE myfriends[
<!ELEMENT myfriends ANY>
<!ELEMENT friendlink ANY>
<!ATTLIST friendlink
    xmlns:xlink CDATA #FIXED "http://www.w3.org/TR/xlink"
    xlink:type (simple|extended|locator|arc) #FIXED "simple"
    xlink:href CDATA #REQUIRED
    xlink:role CDATA #IMPLIED
    xlink:title CDATA #IMPLIED
    xlink:show (new|parsed|replace) "parsed"
    xlink:actuate (user|auto) "auto">
]>
<myfriends>
    <friendlink xmlns:xlink="http://www.w3.org/TR/xlink"
                xlink:href="http://127.0.0.1/personlists.xml">Jane</friendlink>
</myfriends>
```

在定义一个 XLink 元素时，首先声明 XLink 名称空间，即：xmlns：xlink CDATA # FIXED http：//www. w3. org/TR/xlink，该名称空间对应了 XLink 规范中定义的一组元素和属性。利用这些默认属性，来定义用户的 XLink 元素属性，主要的属性包括：

type：指明链接类型是 Simple Link 还是 Extended Link。

href：用来设定链接的地址，与 HTML 中<a>标记的 href 属性一样。

role：声明该链接功能，提供给应用程序读取。

title：声明该链接功能，提供给用户读取，与 HTML 中<a>标记的 alt 属性相似。

show：有三种取值，replace 表示将链接的内容取代当前的内容，new 表示将链接的内容在一个新的窗口打开，embed 表示将链接的内容加入到当前的内容中。

actuate：设置该链接是如何被激活。auto 表示 XML 文件被解读后，链接自动被激活。而 user 表示该链接必须被用户手动激活，也就是用户必须要用鼠标点击一下该链接。

下面通过 XSL 将这个 XML 文件在浏览器中显示出来，即要显示上述 XML 文档中<friendlink>元素所标记的 XML 超链接。friendlists. xsl 代码清单如下：

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                              xmlns:xlink="http://www.w3.org/TR/xlink">
<xsl:output method="html"/>
<xsl:template match="friendlink">
<xsl:if test="@xlink:href">
      <a href="{@xlink:href}"><xsl:value-of select="."/></a>
</xsl:if>
</xsl:template>
</xsl:stylesheet>
```

在浏览器中打开 myfriends.xml 文档,输出结果如图 3-14 所示。




图 3-14　XML 中的超链接

### 2. XPointer 规范

XML 文档是一种结构化文件,这使得借助文件结构进行内部定位成为可能,这就是 XPointer。XPointer 支持在 XML 文件中定位元素、属性、字符串等内部结构,例如,可以定位到根元素或者当前元素的某个子元素,也可以定位到文件中的一个点或两个点之间的区域。

XPointer 基于 XSL 转换中的 XPath 语言,并在其基础上进行了扩展,包括:(1)可以定位结点、点和区域;(2)通过字符串匹配定位资源片段;(3)在 URI 引用中定位资源片断。

由于 XPointer 的功能是文件内部定位,因此它可以使用在需要定位的任何地方,例如在可视化的 XML 编辑器中定位元素、属性等。但人们经常利用 XPointer 描述 XLink 链接的目标资源,因此,通常将 XPointer 和 XLink 放在一起讨论。

XPointer 引用的基本语法为:

```
<linkelementname xlink:href = "xmlDdocument#xpointer(expression)"> text </linkelementname>
```

其中,在 #xpointer(expression)中,表达式 expression 是一个位置路径表达式,和 XPath 中的位置路径相同,用于定位计算,以确定被链接的 XML 文档中的位置。当利用 id("somelocation")进行元素定位时,表示文件中 ID 等于指定常数的元素,可以将 #xpointer(id(name))简写成 #name。

例如,下面是记录个人信息的文档 personslists.xml,定义了多个人的信息,每个人都定义了 id 属性,代码清单如下:

```
<?xml version = "1.0" encoding = "ISO - 8859 - 1"?>
<personlist>
<person id = "Jane">
   <picture url = "http://127.0.0.1/jane.jpg" />
</person>
```

```
<person id = "Cherry">
   <picture url = "http://127.0.0.1/Cherry.jpg" />
</person>
</personlist>
```

在 myfriends.xml 中，修改<friendlink>元素，可以直接定位到一个具体的朋友，例如：

```
<friendlink xmlns:xlink = "http://www.w3.org/TR/xlink"
          xlink:href = "http://127.0.0.1/ personlists.xml#Jane">Jane</friendlink>
```

即可定位到 personlists.xml 文档中的 id 号为 Jane 的<person>结点。

XPointer 中的位置路径在 XPath 的基础上进行了扩展，提供了 5 种不同的在 XML 文件内定位的方法，包括：绝对定位、相对定位、范围定位、属性定位和字符串定位，可将地址定位到相应的地方，功能上比 HTML 中的内链接更为强大，详细介绍略。

### 3.3.9　XML 文档对象模型 DOM 与简单应用程序接口 SAX

W3C 制定了一套书写 XML 分析器的标准接口规范，即 XML 文档对象模型(Document Object Model，DOM)。除此之外，XML_DEV 邮件列表中的成员根据应用的需求也自发地定义了一套对 XML 文档进行操作的接口规范，即 XML 简单应用程序接口(Simple APIs for XML，SAX)。这两种接口规范各有侧重，互有长短，应用都比较广泛。

在 XML 应用程序开发过程中，DOM 和 SAX 所处的地位如图 3-15 所示。

从图中可以看出，应用程序不是直接对 XML 文档进行操作的，而是首先由 XML 分析器对 XML 文档进行分析，然后，应用程序通过 XML 分析器所提供的 DOM 接口或 SAX 接口对分析结果进行操作，从而间接地实现了对 XML 文档的访问。

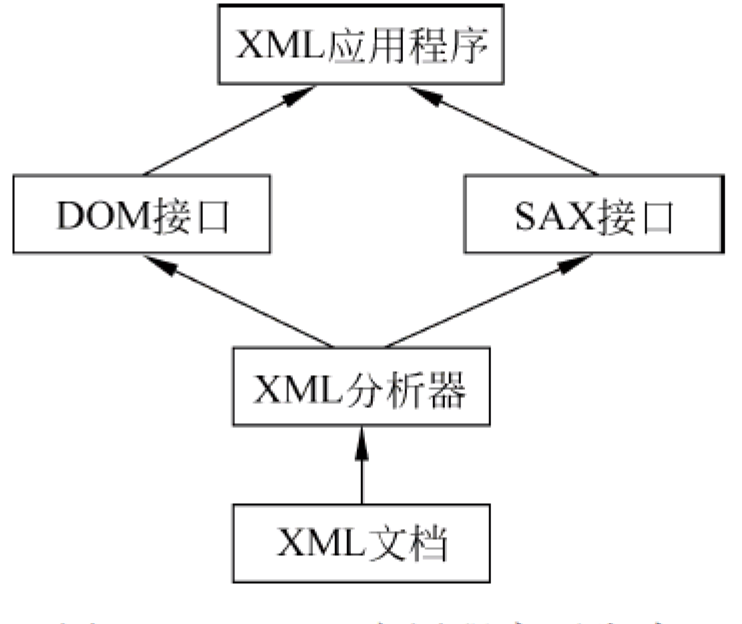



图 3-15　XML 应用程序开发中的 DOM 与 SAX 接口

#### 1. XML 文档对象模型 DOM

XML 文档对象模型(Document Object Model，DOM)是 HTML 文档和 XML 文档的应用程序接口。在应用程序(例如浏览器)中，基于 DOM 的 XML 分析器依据 XML 的文档结构，将一个 XML 文档转换成一棵结点树(通常称 DOM 树)，应用程序通过对这个对象模型的操作，从而实现对 XML 文档数据的操作。

由于 XML 本质上就是一种分层结构，用 DOM 树对 XML 文档进行描述的方法是相当有效的。DOM 树所提供的随机访问方式给应用程序的开发也带来了很大的灵活性，它可以任意地控制整个 XML 文档中的内容。然而，由于 DOM 分析器把整个 XML 文档转化成 DOM 树放在了内存中，因此，当文档比较大或者结构比较复杂时，对内存的需求就比较高。而且，对于结构复杂的树的遍历也是一项耗时的操作。所以，DOM 分析器对机器性能的要求较高，实现效率不十分理想。但是，由于 DOM 分析器所采用的树形结构的思想与 XML 文档的结构相吻合，又可以进行随机访问，因此，DOM 分析器具有广泛的使用价值。

(1) DOM 的组成

对于 XML 应用开发来说,DOM 就是一个对象化的 XML 数据接口,一个与语言无关、与平台无关的标准接口规范。它定义了 HTML 文档和 XML 文档的逻辑结构,给出了一种访问和处理 HTML 文档和 XML 文档的方法。利用 DOM,程序开发人员可以动态地创建文档,遍历文档结构,添加、修改、删除文档内容,改变文档的显示方式等等。可以这样说,文档代表的是数据,而 DOM 则代表了如何去处理这些数据。无论是在浏览器中、在服务器上还是在客户端,只要有用到 XML,都可能用到 DOM。

作为 W3C 的标准接口规范,目前,DOM 由三部分组成,包括:核心(core)、HTML 和 XML。核心部分是结构化文档比较底层对象的集合,定义了一组对象,用于表达 HTML 和 XML 文档中的数据。HTML 接口和 XML 接口两部分则是专为操作具体的 HTML 文档和 XML 文档所提供的高级接口,使对这两类文件的操作更加方便。

(2) 创建 DOM 文档对象和加载 XML

通过 DOM 访问 XML 数据通常有两种方法:第一,使用 C、C++或者 Visual Basic 等访问 XML 数据,此时用户必须安装一些特殊的头文件和库;第二,使用 HTML 和脚本语言 JavaScript、VBScript 等通过 DOM 实现 XML 数据的访问。这里我们只看后者的实现方法。

在 HTML 中访问 XML,首先要创建文档对象。在 JavaScript 中,可通过以下语句完成:

```
var Mydocument = new ActiveXObject("Microsoft.XMLDOM")
```

上述语句在内存中创建一个空对象 Mydocument,要加载一个 XML 文档,需通过 Mydocument 对象的 load 方法完成:

```
Mydocument.async = false
Mydocument.load("XML 文档的 URL")
```

这里我们给 async 属性赋值为 false,表明只有当文档下载完毕,控制才返回给调用进程。load 方法告诉分析器加载指定名字的 XML 文档。XML 文档被加载后,就在内存中形成了一棵 DOM 树。

例如,Books.xml 文档内容如下:

```
<?xml version = "1.0" encoding = "gb2312" ?>
<books>
    <book status = "已售完">
        <title>Web 技术导论</title >
        <author>HaoXingwei</author>
      </book>
    <book status = "热卖中">
        <title> Web 开发技术</title>
        <author> HaoXingwei </author>
    </book>
</books>
```

形成的 DOM 树如图 3-16 所示。

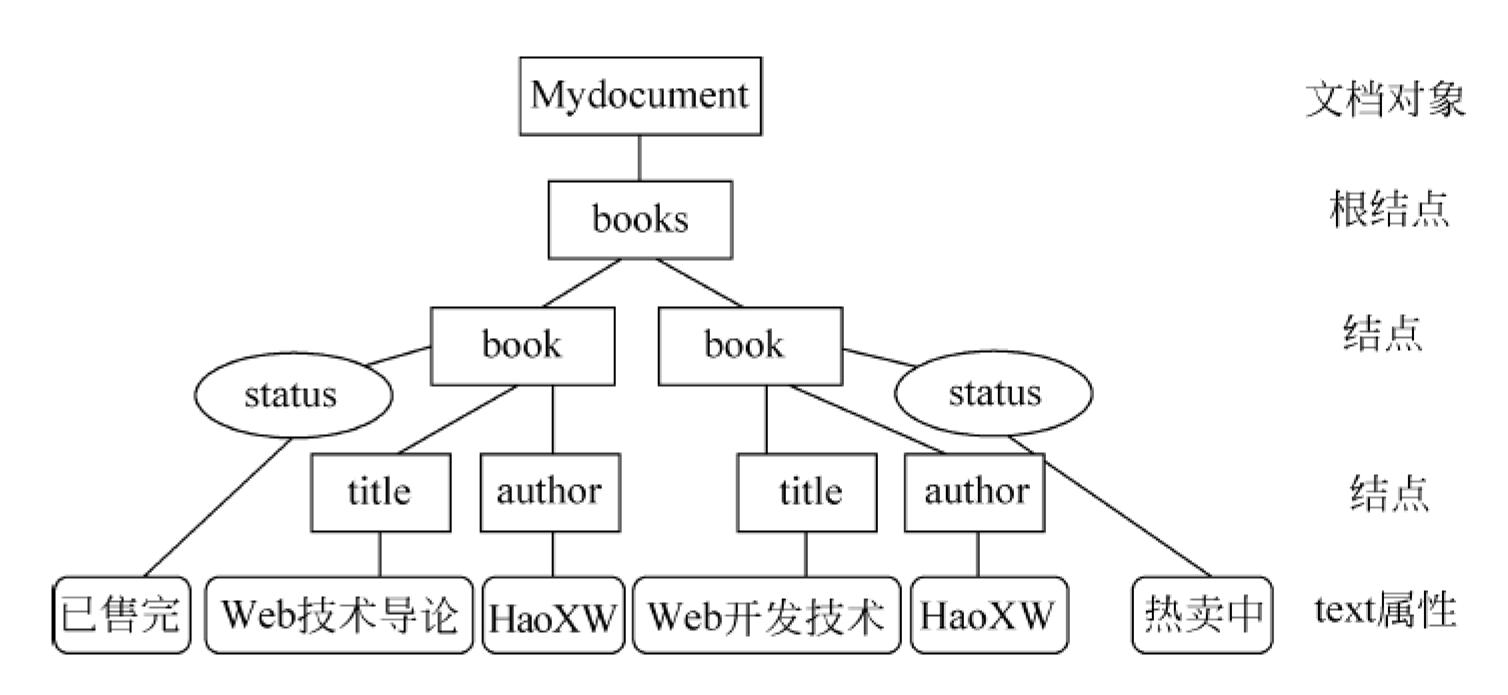



图 3-16 XML DOM 树示意图

从图 3-16 所示的 DOM 树中可以看到，在文档对象中，包含了一个与 XML 文档相一致的树。树的根结点对应于 XML 的根元素，其他结点对应于 XML 中不同层次上的元素。所有结点都是 Node 类型的对象。

(3) DOM 接口对象

文档对象模型利用对象将 XML/HTML 文档模型化，这些模型不仅描述了文档的结构，还定义了模型中对象的行为。在 DOM 树中，每一个结点不是数据结构，而是对象，对象中包含方法和属性。在 DOM 接口规范中，有四个基本的接口：Document、Node、NodeList 和 NamedNodeMap。在这四个基本接口中，Document 接口是对文档进行操作的入口，它是从 Node 接口继承过来的；Node 接口是其他大多数接口的父类，像 Documet、Element、Attribute、Text、Comment 等接口都是从 Node 接口继承过来的；NodeList 接口是一个结点的集合，它包含了某个结点中的所有子结点；NamedNodeMap 接口也是一个结点的集合，通过该接口，可以建立结点名和结点之间的一一映射关系，从而利用结点名可以直接访问特定的结点。

① 文档(Document)对象：Document 对象代表了整个 XML/HTML 文档，是整棵文档树的根，是所有数据、所有其他组件，比如注释和处理指令结点的总的容器，提供了对文档中的数据进行访问和操作的入口。

由于元素、文本结点、注释、处理指令等都不能脱离文档的上下文关系而独立存在，所以在 Document 接口提供了创建其他结点对象的方法，通过该方法创建的结点对象都有一个 ownerDocument 属性，用来表明当前结点是由谁所创建的以及结点同 Document 之间的联系。

② 结点(Node)对象：XML/HTML 中的每一个元素都对应 DOM 树中的一个结点对象。Node 接口在整个 DOM 树中具有举足轻重的地位，DOM 接口中有很大一部分接口是从 Node 接口继承过来的，例如，Element、Attr、CDATASection 等接口。

③ 结点列表(NodeList)对象：这是同一级上的所有结点形成的列表，它包含了多个结点对象。列表中的结点都有与 Sibling 相关联的关系。

在 DOM 中，NodeList 的对象是实时更新(live)的，对文档的改变，会直接反映到相关的 NodeList 对象中。例如，如果通过 DOM 获得一个 NodeList 对象，该对象中包含了某个 Element 结点的所有子结点的集合，那么，当再通过 DOM 对 Element 结点进行操作(添加、删除、改动结点中的子结点)时，这些改变将会自动地反映到 NodeList 对象中，而不需 DOM 应用程序再做其他额外的操作。

NodeList 中的每个 item 都可以通过一个索引来访问，该索引值从 0 开始。

④ 名字空间(NamedNodeMap)对象：包含了可以通过名字来访问的一组结点的集合。但是，和 NodeList 对象不同，NamedNodeMap 所包含的结点集中的结点是无序的。尽管这些结点也可以通过索引来进行访问，但这只是提供了枚举 NamedNodeMap 中所包含结点的一种简单方法，并不表明在 DOM 规范中为 NamedNodeMap 中的结点规定了一种排列顺序。

NamedNodeMap 表示的是一组结点和其唯一名字的一一对应关系，这个接口主要用在属性结点的表示上。与 NodeList 相同，在 DOM 中，NamedNodeMap 对象也是“live”的。

(4) XML DOM 对象的属性和方法

DOM 模型中的对象都有各自的属性集和方法集。通过对象的属性和方法，用户可以操作 XML 文档数据。下面介绍 Document、Node 和 NodeList 三种对象及其属性和方法。

① Document 对象的常用属性和方法

Document 对象的常用属性有：

- documentElement：Element 类型的只读属性，返回文档的根结点对象。
- async：逻辑类型的属性，指定是否允许同步下载文件。
- parseError：返回解析错误对象。

Document 对象的常用方法有：

- load(pathname)：把文件加载到文档对象。
- loadMXL(string)：加载 XML 文档或段。
- createAttribute(name)：创建属性方法。
- createNode(Type，Name，namespaceURI)：创建一个类型为 Type，名称为 Name，并且使用 namespaceURI 作为名称空间的结点。

文档对象还有一些很有用的属性和方法，因篇幅所限不再列出，可从相关的书籍中查到。

② Node 对象的常用属性和方法

Node 对象的常用属性见表 3-7。

**表 3-7 Node 对象常用属性**

| 属性名 | 意　　义 | 属性名 | 意　　义 |
|---|---|---|---|
| atrribute | 结点的属性集 | nextSibling | 当前结点的下一个兄弟结点 |
| childNodes | 当前结点所有子结点的 NodeList | nodeName | 结点名 |
| dataType | 设置或获得结点数据的类型 | nodeType | 结点类型 |
| firstChild | 当前结点的首子结点 | parentNode | 父结点对象 |
| lastChild | 当前结点的末子结点 | text | 设置或获得结点文本 |
| nameSpace | 返回名称空间的 URI | xml | 返回指定结点的 XML 表示 |

Node 对象的常用方法见表 3-8。

**表 3-8 Node 对象常用方法**

| 方法名 | 功　　能 | 方法名 | 功　　能 |
|---|---|---|---|
| appendChild | 添加新结点 | removeChild | 删除子结点 |
| cloneNode | 复制结点 | replaceChild | 替换结点 |
| hasChildNodes | 当前结点是否有子结点 | selectNodes | 选择结点列表及其后代 |
| insertBefore | 插入结点 | selectSingleNode | 选择结点及其后代 |
| parsed | 结点是否被解析 | | |

③ NodeList 对象的属性和方法

NodeList 实际上是一个结点对象列表，它只有一个 Length 属性，其值为对象中包含结点的个数。方法有：item(Index)、nextNode()和 reset()三个，详细介绍略。

**【例 3-16】** 对 XML 文档“book. xml”，在 HTML 中通过 DOM 模型将各本书的书名加到一个列表控件中，单击列表控件中的书名，显示该书数据。

**清单 1：** book. xml 文档内容

```
<?xml version = "1.0" encoding = "gb2312" ?>
<中国古典名著>
  <书>
    <书名>三国演义</书名>
    <作者>罗贯中</作者>
    <故事梗概>略…</故事梗概>
  </书>
<书>
    <书名>西游记</书名>
    <作者>吴承恩</作者>
    <故事梗概>略…</故事梗概>
</书>
<书>
    <书名>红楼梦</书名>
    <作者>曹雪芹</作者>
    <故事梗概>略…</故事梗概>
  </书>
<书>
  <书名>水浒传</书名>
  <作者>施耐庵</作者>
  <故事梗概>略…</故事梗概>
</书>
</中国古典名著>
```

**清单 2：** book. htm 文档内容

```
<html>
<head>
<title>The Chinese famous book</title>
<script language = "JavaScript">
function selectbook(i)
{
  booknode = root.childNodes.item(i);
  booktitle.innerText  = booknode.childNodes.item(0).text;
  bookauthor.innerText = booknode.childNodes.item(1).text;
  bookbrief.value      = booknode.childNodes.item(2).text;
}
</script>
</head>

<body>
<h1 align = "center"><font size = "5" color = "#0000FF">中国古典名著</font></h1>
```

```
<center>
<table border = "1" width = "400" bgcolor = "#CCFFCC">
 <tr>
  <td width = "100" height = "30" align = "center">书 名</td>
  <td width = "270" height = "30" align = "center">简要介绍</td>
</tr>
 <tr>
    <td width = "100" valign = "top" align = "left" >
      <select size = "4" id = "D1" onChange = "selectbook(D1.selectedIndex)">
        <script language = "JavaScript">
          var mybook = new ActiveXObject("Microsoft.XMLDOM");
          mybook.async = false;
          mybook.load("book.xml");
          root = mybook.documentElement;
          for (var i = 0; i<root.childNodes.length; i++)
          {
            booknode = root.childNodes.item(i);
            a1 = booknode.childNodes.item(0).text;
            document.write("<OPTION>");
            document.write(a1);
            document.writeln("</OPTION>");
          }
        </script>
      </select>
    </td>
    <td width = "270" align = "left">
      <ul><li>书名：<span id = "booktitle"></span></li>
            <li>作者：<span id = "bookauthor"></span></li>
            <li>故事梗概：</li>
            <textarea rows = "4" name = "bookbrief" cols = "30"></textarea>
      </ul>
    </td>
  </tr>
</table>
</center>
</body>
</html>
```

在上述 book.htm 代码中，<span id="booktitle"></span>和<span id="bookauthor"></span>起一个占位符的作用，以便通过程序可以修改该处的显示内容。保存文件，在浏览器中打开以上 book.htm 文档，可以看到显示的页面，当点击左侧列表控件中的书名时，在右侧的列表中可以显示出该书的所有数据，文档 book.htm 在 IE 浏览器中的显示结果如图 3-17 所示。

不是所有的浏览器均支持 Microsoft.XMLDOM，Firefox 浏览器将不能显示上述页面。最后我们对这个例子做一个总结，通过上面的这个例子，我们可以看出 XML 文档是对数据的一种描述，是存储数据的一种形式，而 HTML 文档的重点在于展示数据，理解这一点对于理解 XML 是非常重要的。




图 3-17　应用 DOM 对象示例

### 2. XML 简单应用程序接口 SAX

XML 简单应用程序接口(Simple APIs for XML,SAX)与 DOM 不同,SAX 提供的访问模式是一种顺序模式,这是一种快速读写 XML 数据的方式。当使用 SAX 分析器对 XML 文档进行分析时,会触发一系列事件,并激活相应的事件处理函数,应用程序通过这些事件处理函数实现对 XML 文档的访问,因而 SAX 接口也被称作事件驱动接口。

SAX 分析器在对 XML 文档进行分析时,会触发一系列事件,由于事件触发本身是有时序性的,因此,SAX 提供的是一种顺序访问机制,对于已经分析过的部分,不能再倒回去重新处理。SAX 分析器在实现时,它只是顺序地检查 XML 文档中的字节流,判断当前字节是 XML 语法中的哪一部分,是否符合 XML 语法,然后再触发相应的事件,而事件处理函数本身则要由应用程序自己来实现。同 DOM 分析器相比,SAX 分析器缺乏灵活性。然而,由于 SAX 分析器实现简单,对内存要求比较低,因此实现效率比较高,对于那些只需要访问 XML 文档中的数据而不对文档进行更改的应用程序来说,SAX 分析器更为合适。

SAX 分析器的基本构成框架如图 3-18 所示。

SAXParserFactory 用来生成一个分析器实例。XML 文档从左侧箭头所示处读入,当分析器对文档进行分析时,会触发在 DocumentHandler、ErrorHandler、DTDHandler 以及 EntityResolver 接口中定义的回调方法。用户通过编写各事件处理函数实现对文档的访问处理,详细介绍请参考其他专门教材。

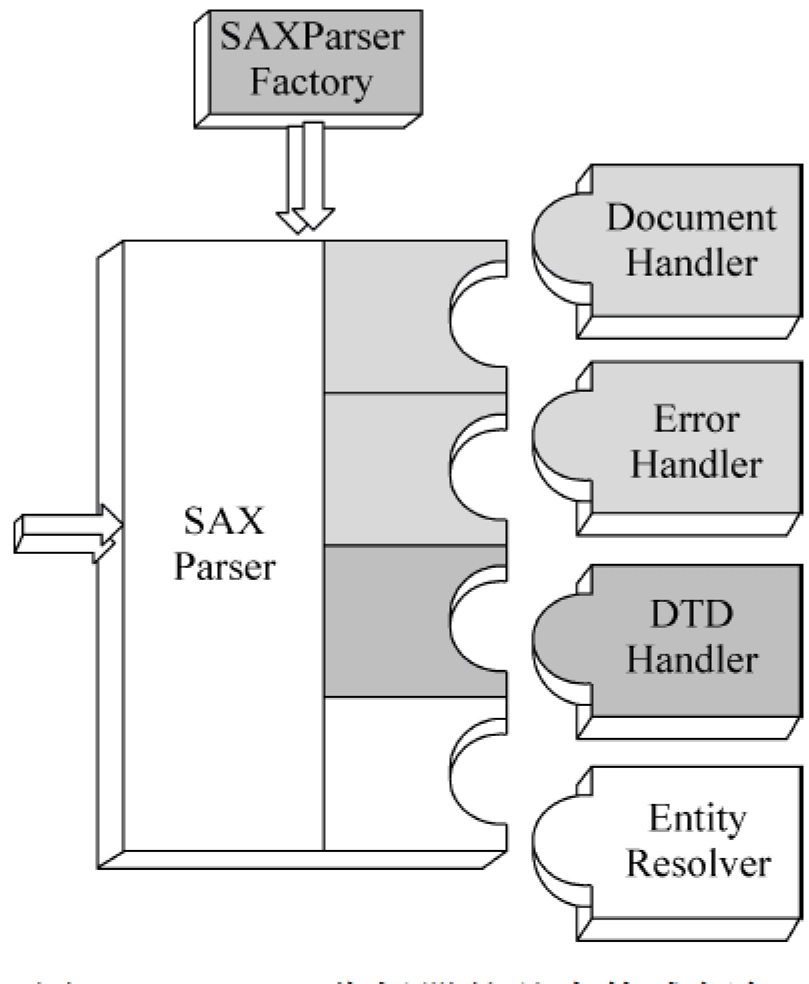



图 3-18　SAX 分析器的基本构成框架

## 3.4 XML 开发环境 XMLSpy

XML 文档是一种纯文本文档,我们可以用“记事本”、UltraEdit 等文本编辑软件来编辑,但是非常麻烦。Altova XMLSpy 是一个可视化的 XML 编辑和开发环境,专门用于设计、编辑和调试企业级的基于 XML 技术的应用,包括 XML、XML Schema、XSL/XSLT、SOAP、WSDL 和互联网服务技术,同时也是 J2EE、.NET 和数据库开发人员不可缺少的高性能的开发工具,可以大大提高应用系统的开发效率。

### 3.4.1 XMLSpy 简介

Altova XMLSpy 是工业标准的 XML 开发环境,XMLSpy 支持下列功能:

(1) 创建并编辑 XML 实例文档。
(2) DTD 编辑。
(3) XML Schema 开发。
(4) WSDL 开发。
(5) SOAP 开发和调试。
(6) XSLT 开发和调试。
(7) XPath 开发。
(8) XQuery 开发和调试。
(9) 数据库交互。
(10) Web 服务开发。
(11) Java/C#/C++代码生成。
(12) VS.NET 和 Eclipse 集成。
(13) 工程管理。

Altova XMLSpy 有多个不同的版本,其最新版为 Altova XMLSpy 2008,分为企业版和专业版两种版本。专业版提供一个 XML 开发环境,包括 XML 文档编辑,架构设计,文件转换与调试。支持 XSLT、XQuery、Databases、VS.NET 和 Eclipse 的集成等。企业版还具有自动代码生成、支持 Web 服务等功能。

关于 Altova XMLSpy 的详细信息请参考其官方网站:http://www.xmlspy.com/,可以下载软件试用版。本书将以 Altova XMLSpy 2006 企业版中文版为例,简要介绍 XMLSpy 的使用方法。

### 3.4.2 XMLSpy 基础

首先安装 XMLSpy 2006 企业版,运行 XMLSpy,显示 XMLSpy 2006 企业版主界面,如图 3-19 所示。

XMLSpy 支持 XML、XSL、XSLT、DTD、Schema 等多种文件格式的编辑器。它可以将 XML 展示为完美的树形结构,可以方便地使用各种 HTML/XML/XSLT 标记,使用它可

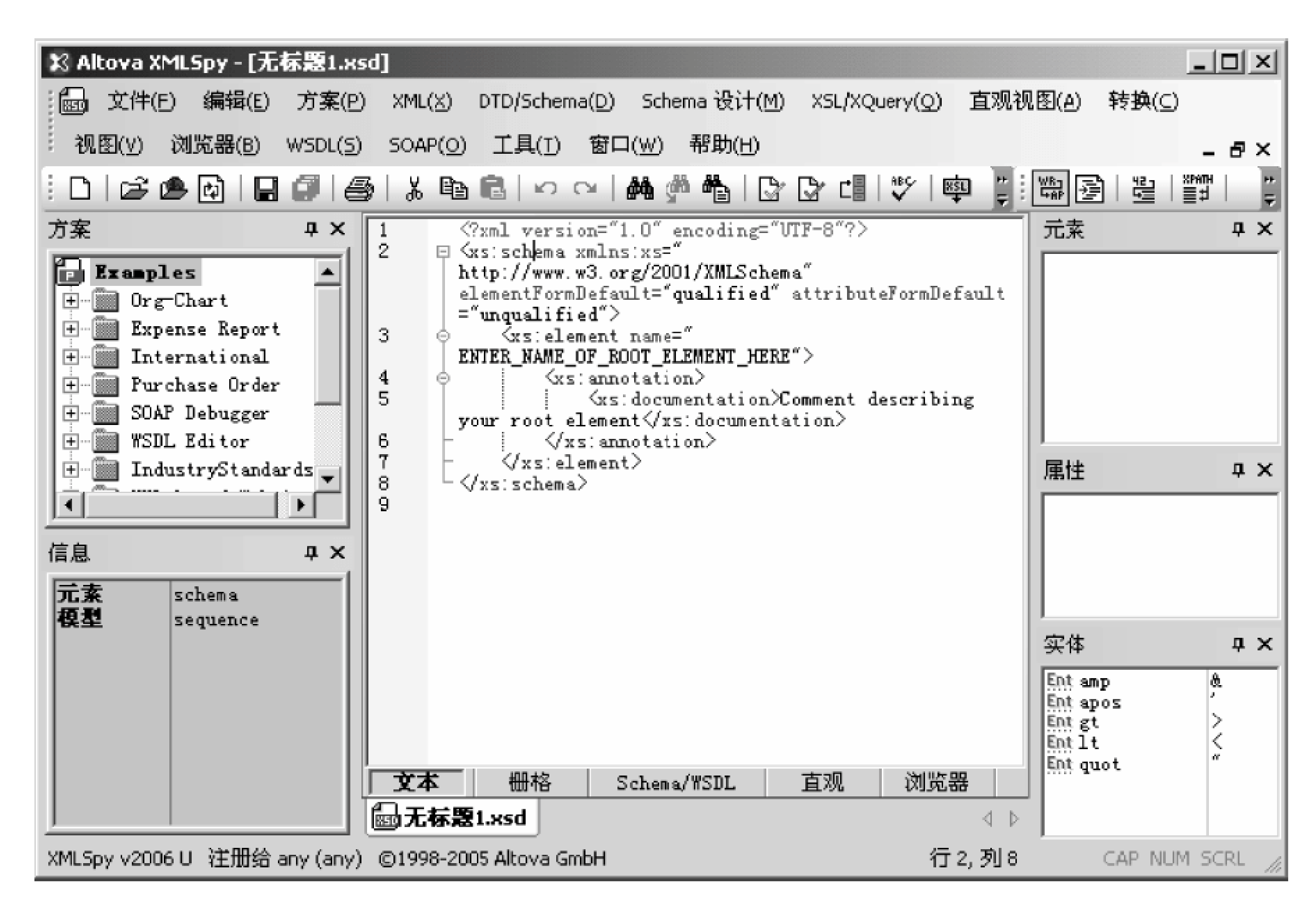


图 3-19　XMLSpy 2006 企业版中文主界面

以大大节约开发时间,不必把大量的时间浪费在代码的输入上。下面通过一个存储电影信息的实例来学习 XMLSpy 的简单使用方法。

首先,设计三个文件:films. xml、film. dtd 和 film. xslt。films. xml 负责存储具体电影内容数据,film. dtd 负责对 films. xml 的验证,而 film. xslt 则负责对 films. xml 进行样式变换,确定它在浏览器里的最终显示效果。

(1) film. dtd 代码清单

```
<?xml version="1.0" encoding="GB2312"?>
<!ELEMENT movies (id,name,time)>
<!ATTLIST movies type CDATA #REQUIRED>
<!ELEMENT id (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT time (#PCDATA)>
```

(2) film. xslt 代码清单

```
<?xml version="1.0" encoding="gb2312"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="xml" version="1.0" encoding="GB2312" indent="yes"/>
<xsl:template match="/">
<html>
<head>
<title>使用 xslt 示例</title>
</head>
<body>
<xsl:apply-templates></xsl:apply-templates>
</body>
```

```
</html>
</xsl:template>
<xsl:template match="movies">
<p align=center>巩俐代表作</p>
<table border="1" width="400" bgcolor="#CCFFCC">
<tr>
<td>名称</td>
<td>时间</td>
</tr>
<tr>
<td><xsl:value-of select="name"></xsl:value-of></td>
<td><xsl:value-of select="time"></xsl:value-of></td>
</tr>
</table>
</xsl:template>
</xsl:stylesheet>
```

(3) films.xml 代码清单

```
<?xml version="1.0" encoding="GB2312"?>
<!DOCTYPE movies SYSTEM "film.dtd">
<?xml-stylesheet type="text/xsl" href="film.xslt"?>
<movies type="代表作">
<id>1</id>
<name>红高粱</name>
<time>1987</time>
</movies>
```

### 1. 使用 XMLSpy 建立 saveit.dtd 文档

(1) 建立根结点 movies

选择菜单"文件/新建…"命令,打开"创建新文档"对话框,如图 3-20 所示。




图 3-20 XMLSpy"创建新文档"对话框

在创建文档类型列表中,选择"dtd Document Tpye Definition"文件类型,创建一个空的 DTD 文档并在编辑区打开,如图 3-21 所示。

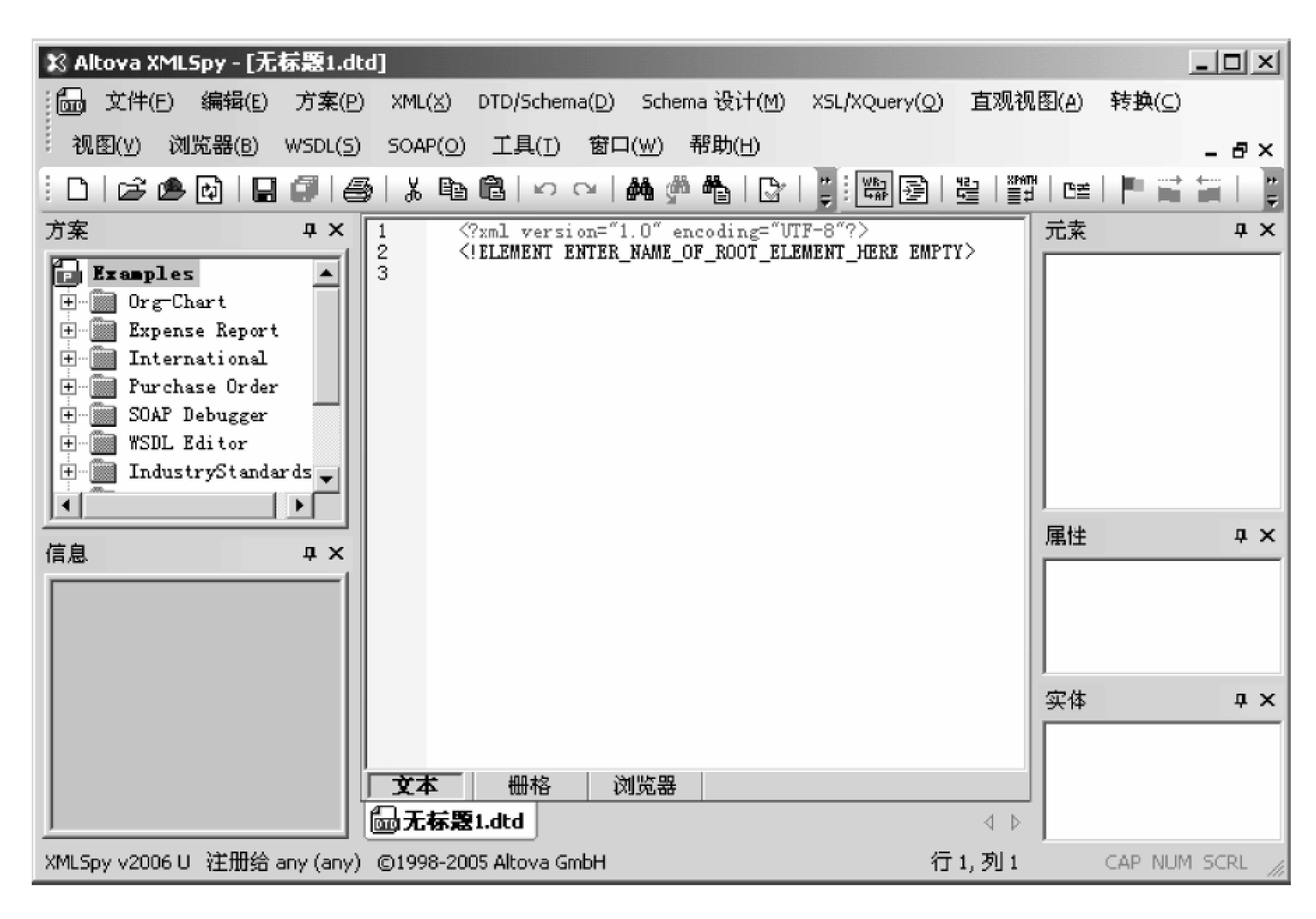


图 3-21　dtd 文档编辑界面

修改编码方式项 enconding 默认值 UTF-8，将其改为 GB2312。

在编辑区，选择“栅格”，在 EIM 处双击，输入 movies。完成后如图 3-22 所示。



图 3-22　dtd 文档编辑界面栅格视图

保持 EIM movies 的选中状态，双击右侧 Elements 框里的 sequence of，结果如图 3-23 所示。

这样根结点 movies 就建立完毕了。




图 3-23　根结点 sequence 属性

(2) 为根结点 movies 添加子结点 id、name、time 及属性 type

在 movies sequence of 上右击鼠标，在快捷菜单中，执行“添加子结点/元素”命令，为其增加三个子结点。

再次在 movies sequence of 上右击鼠标，执行“添加/元素列表”命令，设置其名称为 movies，设置 Name 为 type，设置 Type 为 CDATA(从元素列表中选择)，设置 Presence 为 #REQUIRED(从元素列表中选择)。如图 3-24 所示。

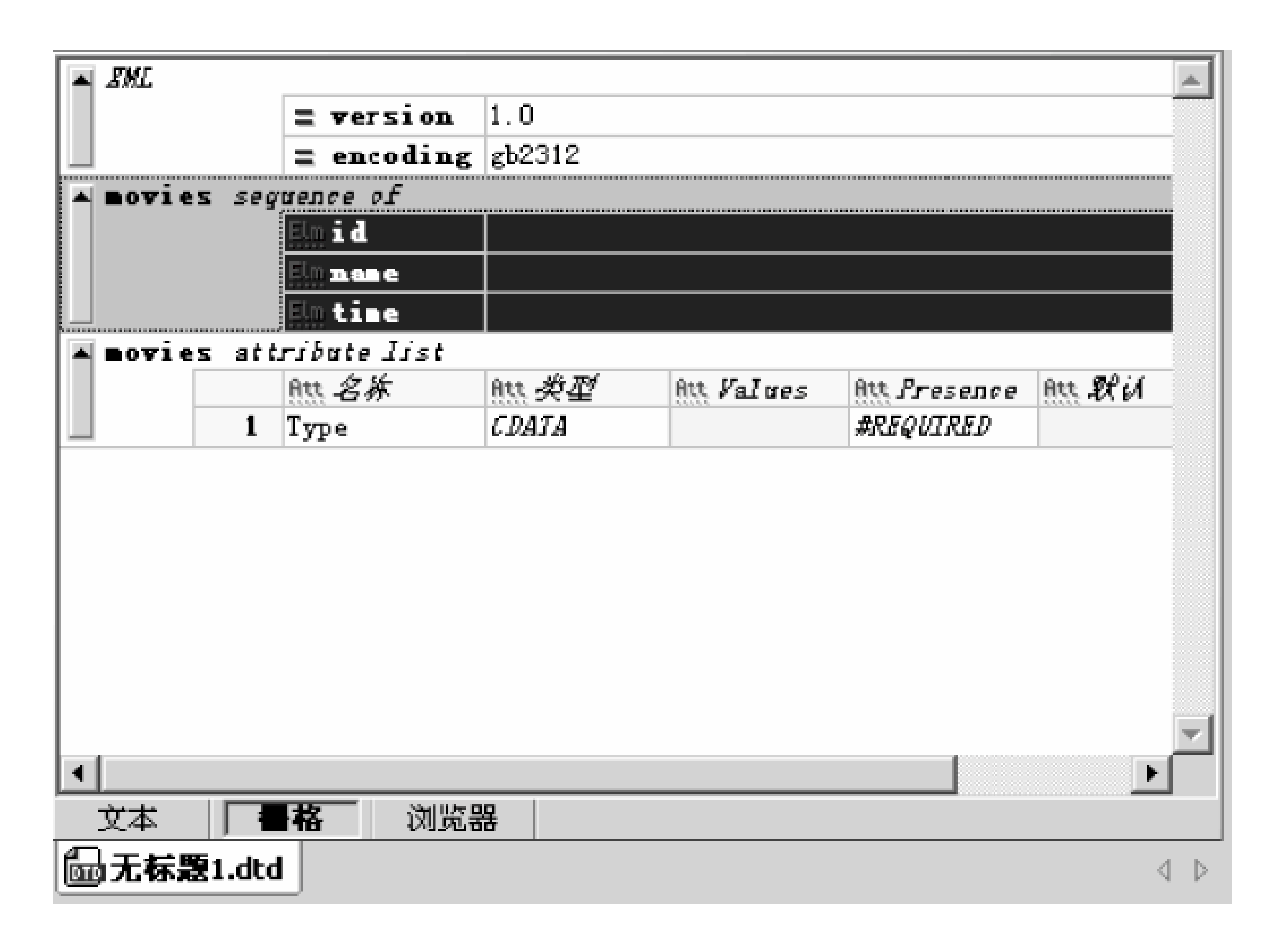



图 3-24　为容器元素类型添加子元素和属性

(3) 建立子结点 id、name、time

在 movies sequence of 上右击鼠标，选择“添加/元素”命令，增加三个结点，数据类型全部设为 PCDATA。如图 3-25 所示。

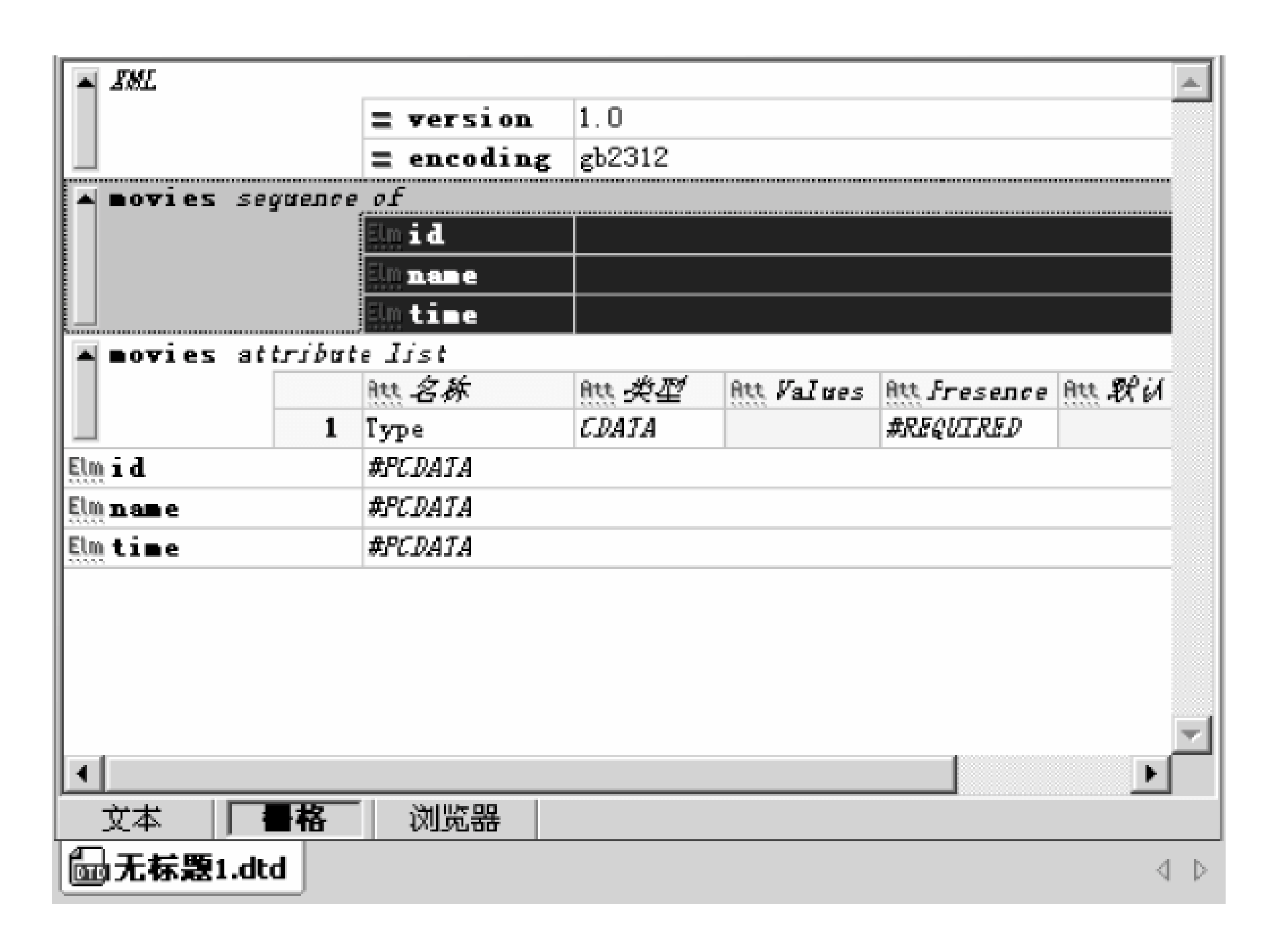



图 3-25　添加简单元素

这样 DTD 文档就建立好了。

执行"文件/保存"命令，将文件命名为 saveit. dtd，保存到 d:\xmlspy 目录下。

在文本视图下可以查看编辑所得到的源代码，如图 3-26 所示。




图 3-26　编辑完成后生成的 dtd 代码

## 2. 使用 XMLSpy 建立 filmxslt 文档

（1）选择菜单"文件/新建…"命令，打开"创建新文档"对话框，选择里面最后一项 xslt (Extensible Stylesheet Language)项，建立的新 XSLT 文件如图 3-27 所示。

此时 XMLSpy 会将视图自动转到代码编辑视图下，因为直接编辑 XSLT 更为方便一些。将其编码方式改为 GB2312。

（2）其余的代码可以通过图 3-28 所示的"元素"面板方便地添加到编辑区中。

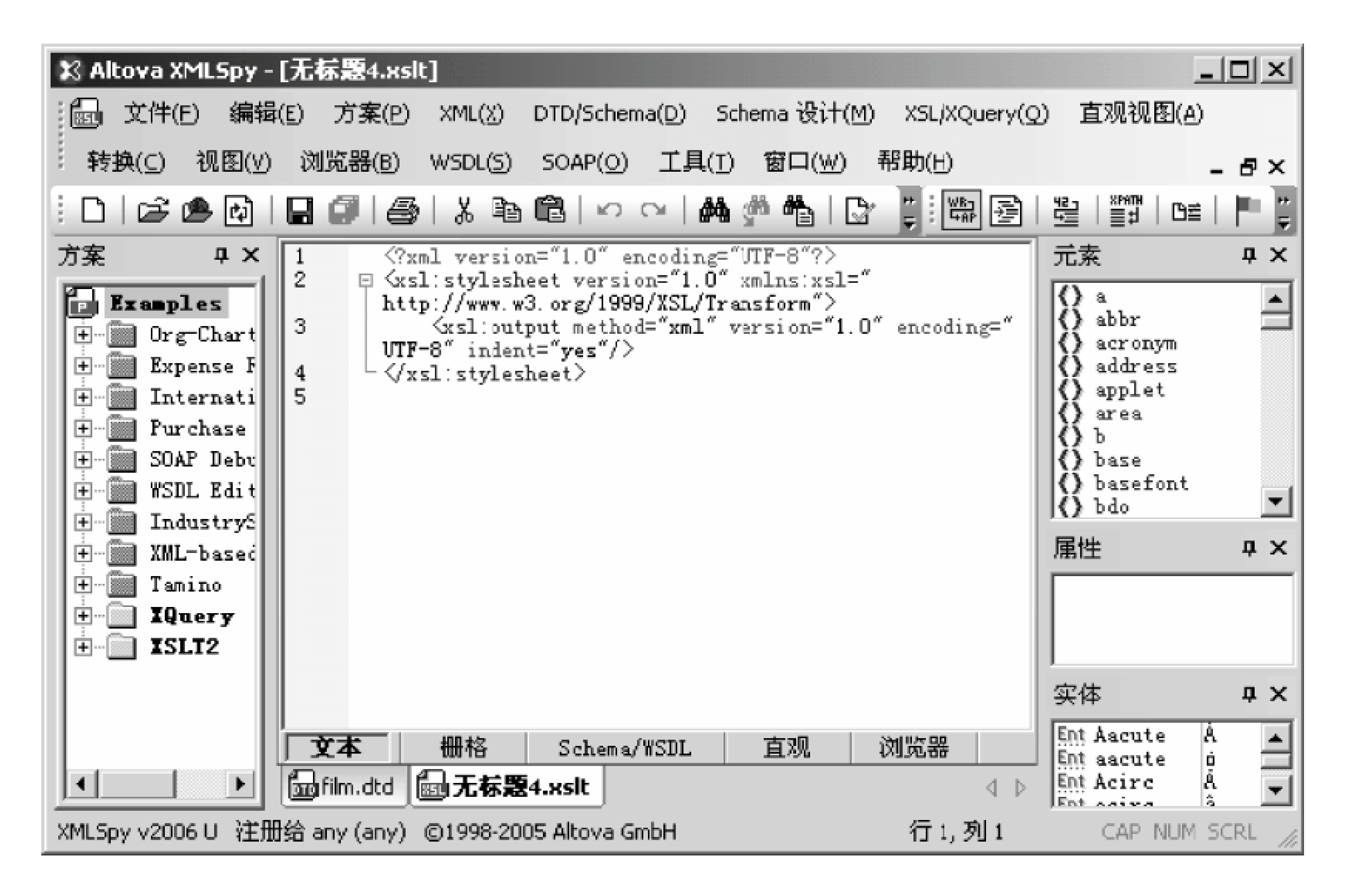



图 3-27 编辑 XSLT 文档

```
<?xml version="1.0" encoding="GB2312"?>
<xsl:stylesheet version="1.0" xmlns:xsl="
http://www.w3.org/1999/XSL/Transform">
    <xsl:output method="xml" version="1.0" encoding="gb2312"
 indent="yes"/>
    <xsl:template match="/">
        <html>
            <head>
                <title>使用xslt示例</title>
            </head>
            <body>
                <xsl:apply-templates/>
            </body>
        </html>
    </xsl:template>
    <xsl:template match="movies">
        <p align="center">巩俐代表作</p>
        <table border="1" width="400" bgcolor="#CCFFCC">
            <tr>
                <td>名称</td>
                <td>时间</td>
            </tr>
            <tr>
                <td>
                    <xsl:value-of select="name"/>
                </td>
                <td>
                    <xsl:value-of select="time"/>
                </td>
            </tr>
        </table>
    </xsl:template>
</xsl:stylesheet>
```

文本 栅格 Schema/WSDL 直观 浏览器

film.dtd 无标题4.xslt

图 3-28 XSLT 文件

添加完毕,设置相应的结点为各元素的属性值即可完成 XSLT 文档的编写。执行“文件/保存”命令,保存到 d:\:xmlspy 目录下,文件名为 film.xslt。

### 3. 使用 XMLSpy 建立 films.xml 文档

(1) 选择菜单“文件/新建…”命令,打开“创建新文档”对话框,选择里面的 xml(XML Document),此时会弹出一个对话框,要求选择 XML 文档的验证方式是 DTD 还是 Schema,

如图 3-29 所示。

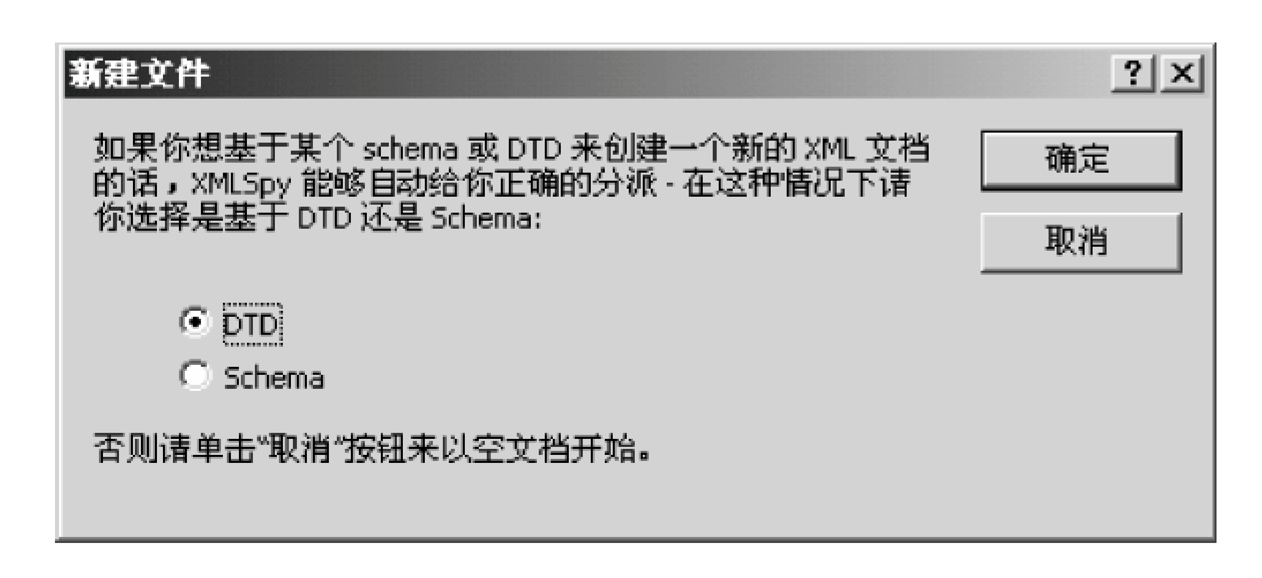



图 3-29　选择验证类型

选择 DTD 验证方式，单击“确定”按钮，然后选择刚刚创建的 film. dtd 作为其验证文档，单击“确定”按钮。

(2) XMLSpy 将自动建立一个符合 film. dtd 验证的 XML 空白文档。填入内容数据。将编码方式项 enconding 更改为 GB2312。

如果使用的是汉化版 XMLSpy，可能因为汉化导致一些字符串解析出现问题，比如“SYSTEM 解析成系统”，从而导致系统提示“该文件不具有良好格式：DOCTYPE-ExternalID 的名称必须既是 SYSTEM，又是 PUBLIC”。如图 3-30 所示。

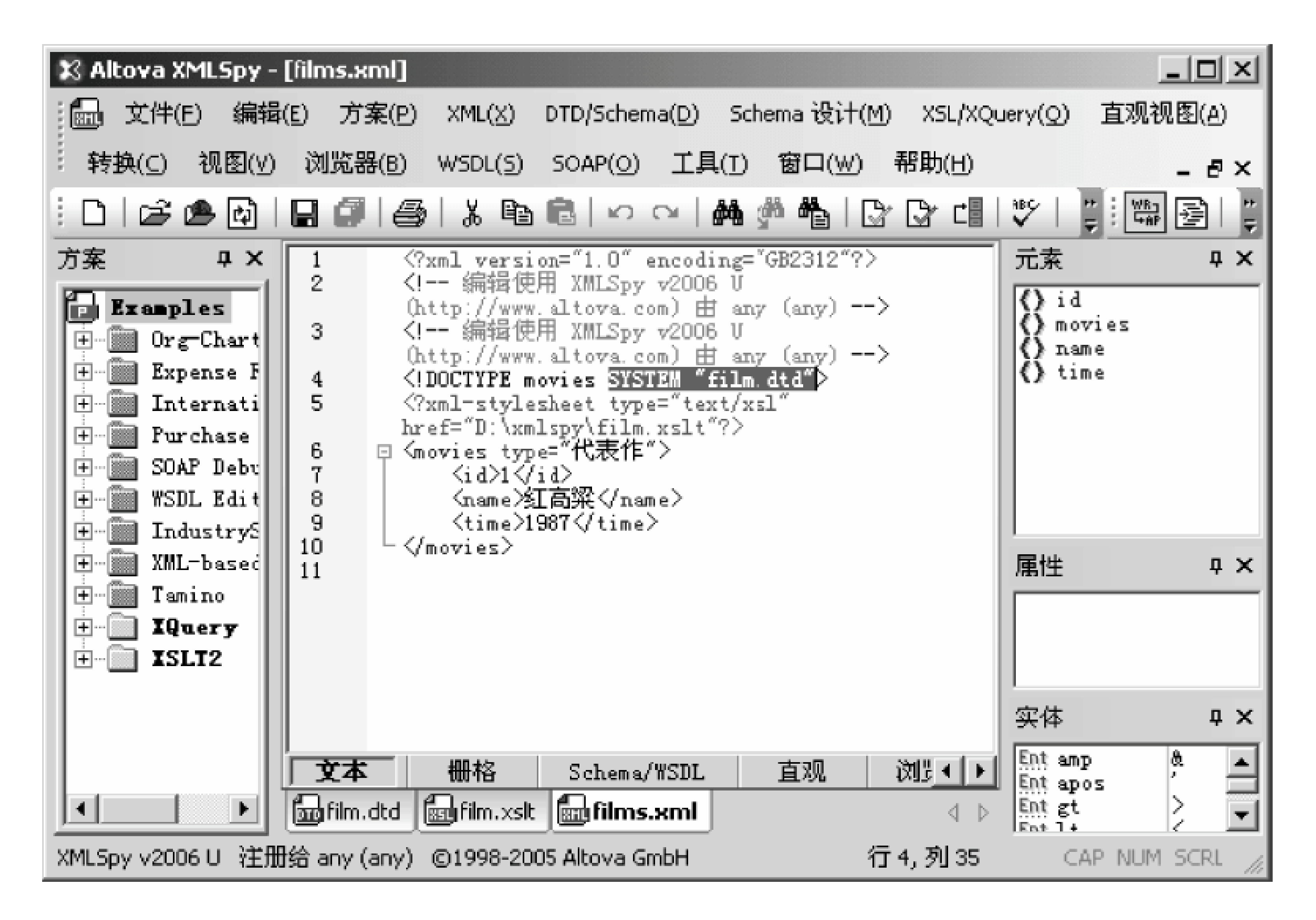


图 3-30　创建 XML 实例文档

如果是英文版就没有问题了，强行保存，在浏览器中打开没有问题。

(3) 执行“XSL/指定 XSL”菜单，并在弹出窗口中选择 d:\xmlspy\film. xslt 文件，然后单击“确定”按钮，如图 3-31 所示。

(4) XML 文档终于编辑完毕，存盘到 d:\xmlspy 目录下，命名为 films. xml。

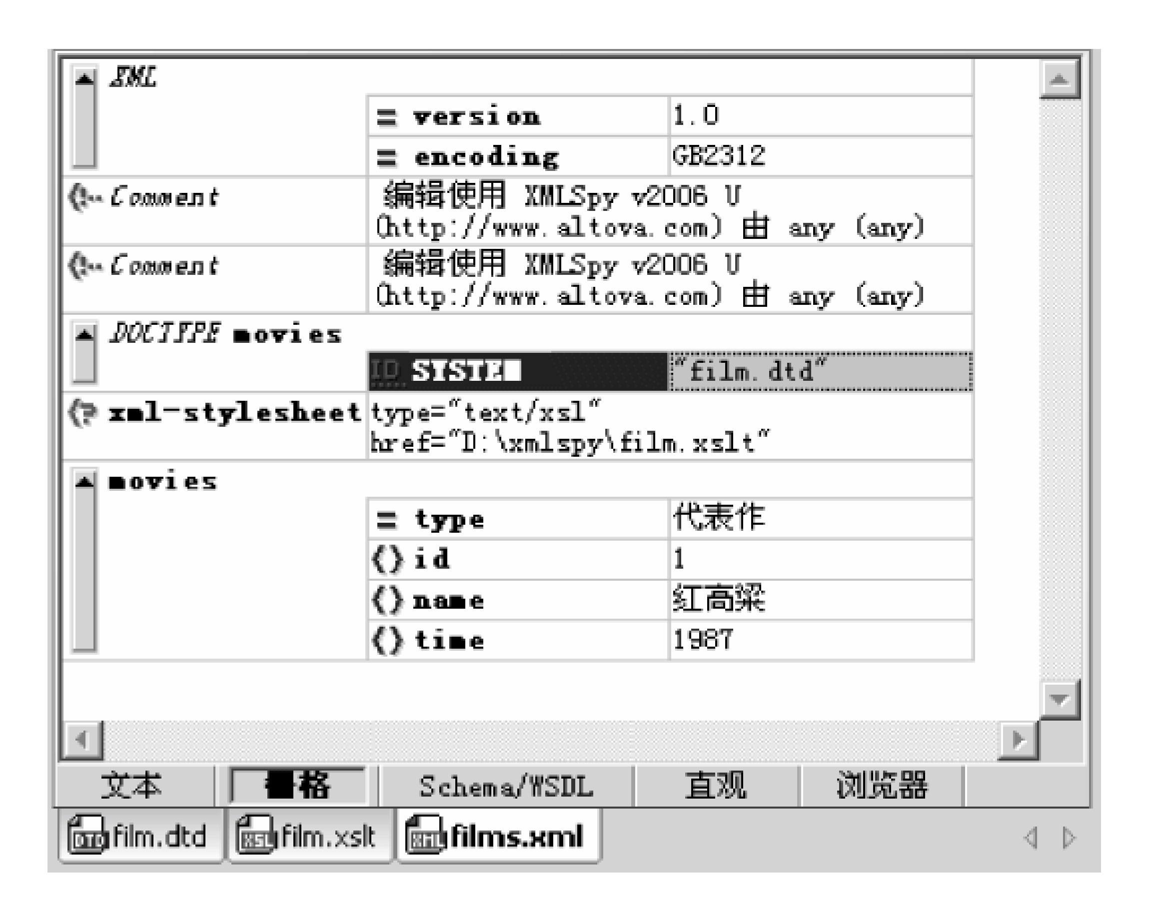



图 3-31 编辑后的 XML 实例文档

### 4. 查看 films.xml 显示效果

选择“XSLT->XSL Transformation”或点击来直接在 XMLSpy 中查看 films.xml 的最终显示效果。也可以到 d:\xmlspy 目录下使用浏览器观看,但浏览器必须是 IE6 以上的版本。如果想输出变换结果文档,可以在 XMLSpy 中变换后点击将结果文档存盘即可。

最终显示效果如图 3-32 所示。




图 3-32 films.xml 实例文档在浏览器中的显示界面

## 3.4.3 系统建模与数据验证

XML Schema 可以定义元素及其类型,通过 Schema 可以构建数据模型,来约束数据存储和交换,以及实现数据的合法性校验。我们可能注意到,对于一个 XML 文档(可以看作是一个 XML Schema 的实例),即使它引用了一个 Schema,当在浏览器打开这个 XML 文档时,其存储的数据和 Schema 不一致,也不会报错,这似乎不符合 XML Schema 最初的设计思想。为什么呢?因为浏览器只检查 XML 文档的格式,并不进行数据合法性的检查。要

检查 XML 文档数据的合法性，需要使用 XMLSpy。

下面通过一个例子说明 XML 文档实例数据合法性的检验。

**【例 3-17】** 使用 XML Schema 进行系统建模与 XML 文档数据合法性检验。

首先看下面两段代码：

(1) 描述一本书的 XML Schema，代码清单如下(文件名：books. xsd)：

```
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema">
<xsd:element name = "catalog" type = "CatalogData"/>
<xsd:complexType name = "CatalogData">
     <xsd:sequence>
          <xsd:element name = "book" type = "bookdata" minOccurs = "0" maxOccurs = "unbounded"/>
     </xsd:sequence>
</xsd:complexType>
<xsd:complexType name = "bookdata">
     <xsd:sequence>
          <xsd:element name = "title" type = "xsd:string"/>
          <xsd:element name = "author" type = "xsd:string"/>
          <xsd:element name = "price" type = "xsd:float"/>
          <xsd:element name = "description" type = "xsd:string"/>
          <xsd:element name = "publish_date" type = "xsd:date"/>
     </xsd:sequence>
     <xsd:attribute name = "id" type = "xsd:string"/>
</xsd:complexType>
</xsd:schema>
```

(2) 一个 XML 实例文档，代码清单如下(文件名：mybooks. xml)：

```
<?xml version = "1.0"?>
<x:catalog xmlns:x = "urn:books.xsd">
<book id = "ISBNxxxxx">
   <title>Web Developing Technologies Guide</title>
   <author>Hao XW</author>
   <price>28.50</price>
   <description >An widely in - depth look and talk on creating applications on WEB</ description>
   <publish_date>2006 - 05 - 16</publish_date>
</book>
</x:catalog>
```

最后，需要说明的是，XML Schema 通常保存的扩展名是. xsd(XML Schema Defination)。

使用 XMLSpy 进行系统建模与 XML 文档数据合法性检验过程如下：

#### 1. 创建 XML Schema 模式文件 book.xsd

在 XMLSpy 中，执行“文件/新建…”菜单命令，打开“创建新文档”对话框，从文件类型列表中选择“xsd W3C XML Schema”文件类型，创建一个新的 XSD 文档。如图 3-33 所示。

将代码清单 book. xsd 复制到编辑区，或者按照代码清单的内容一步步地建立 book. xsd。如图 3-34 所示。

执行“文件/保存”命令，将 xsd 文件保存到 D:\xmlspy 文件夹下，文件名为 book. xsd。

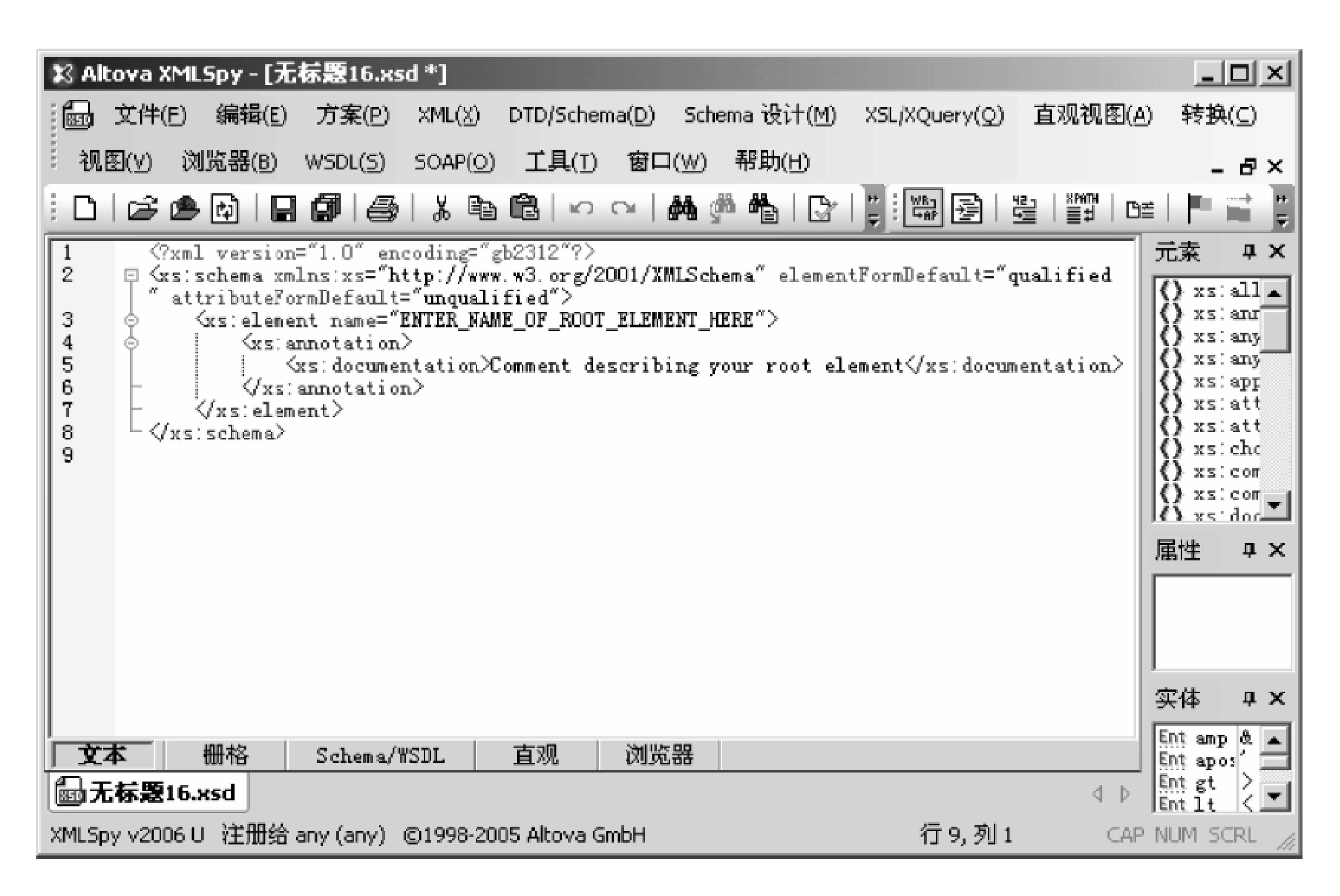


图 3-33 新建 XML Schema 文档

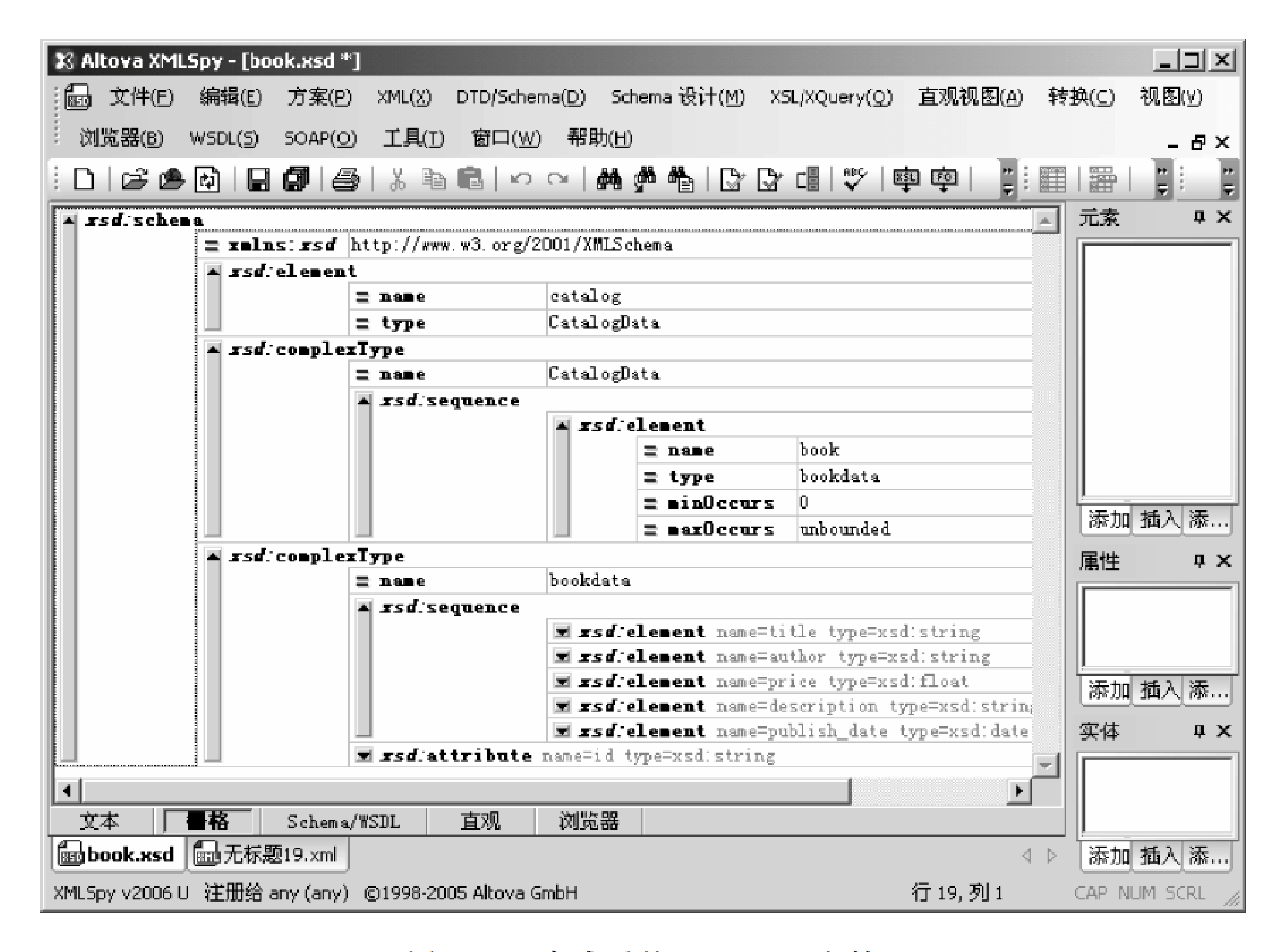


图 3-34 完成后的 book. xsd 文档

## 2. 创建 XML 实例文档 mybooks.xml

执行"文件/新建…"菜单命令，打开"创建新文档"对话框，从文件类型列表中选择"XML Document"文件类型。弹出"新建文件"对话框，如图 3-35 所示。

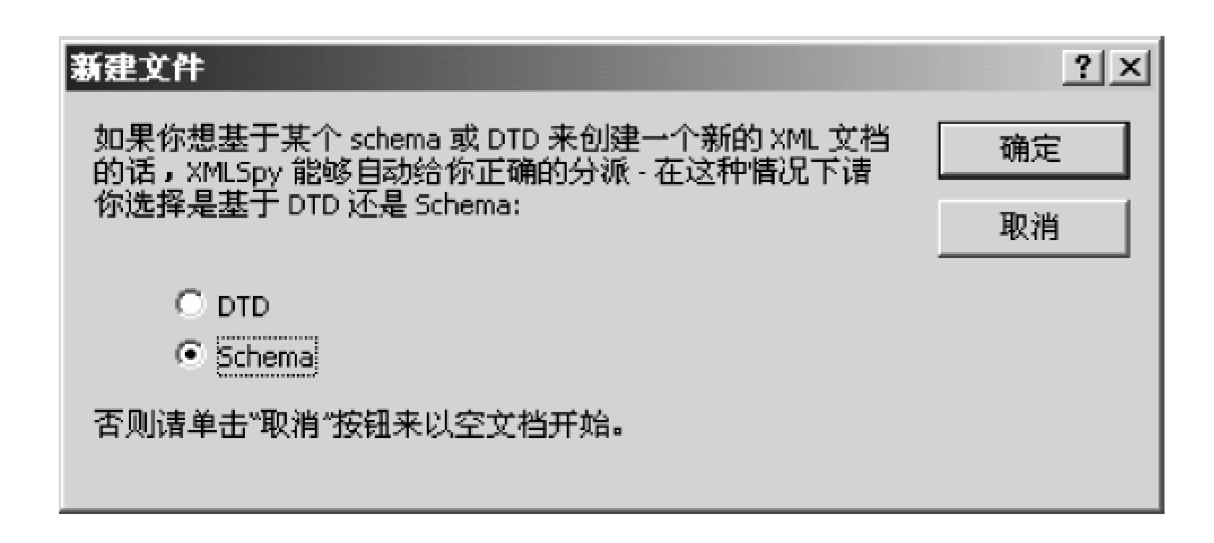



图 3-35　选择验证 Schema

选择 Schema，单击"确定"按钮，如图 3-36 所示。

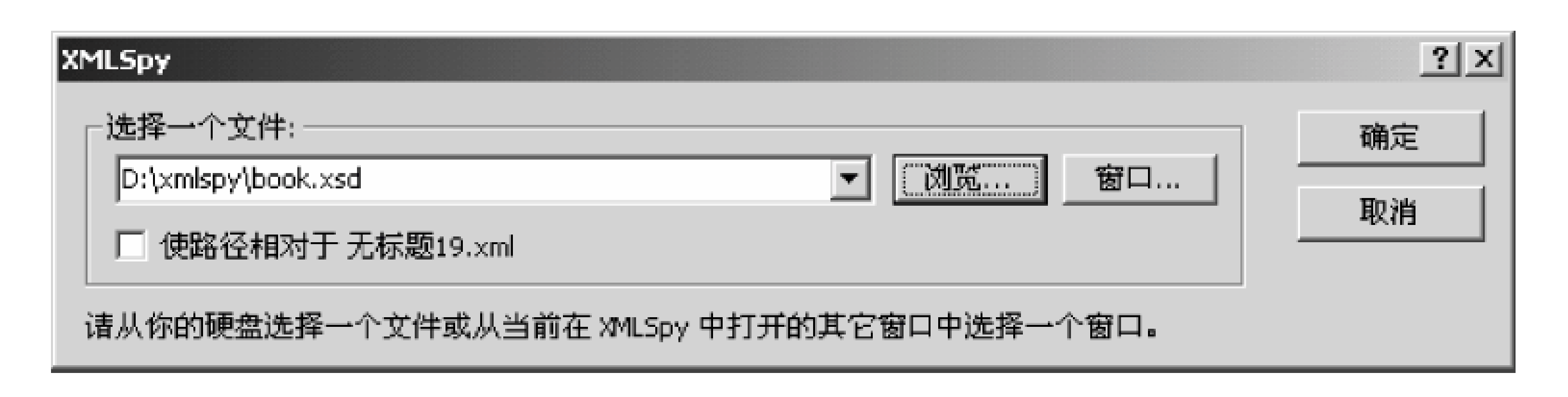



图 3-36　指定模式文件

选择 book. xsd 模式文档，单击"确定"按钮。生成一个新的 XML 文档，包含代码如下：

```
<?xml version = "1.0" encoding = "UTF - 8"?>
<catalog xmlns:xsi = http://www.w3.org/2001/XMLSchema - instance
         xsi:noNamespaceSchemaLocation = "D:\xmlspy\book.xsd"/>
```

在 XML 文档编辑区，选择"栅格"视图，如图 3-37 所示。

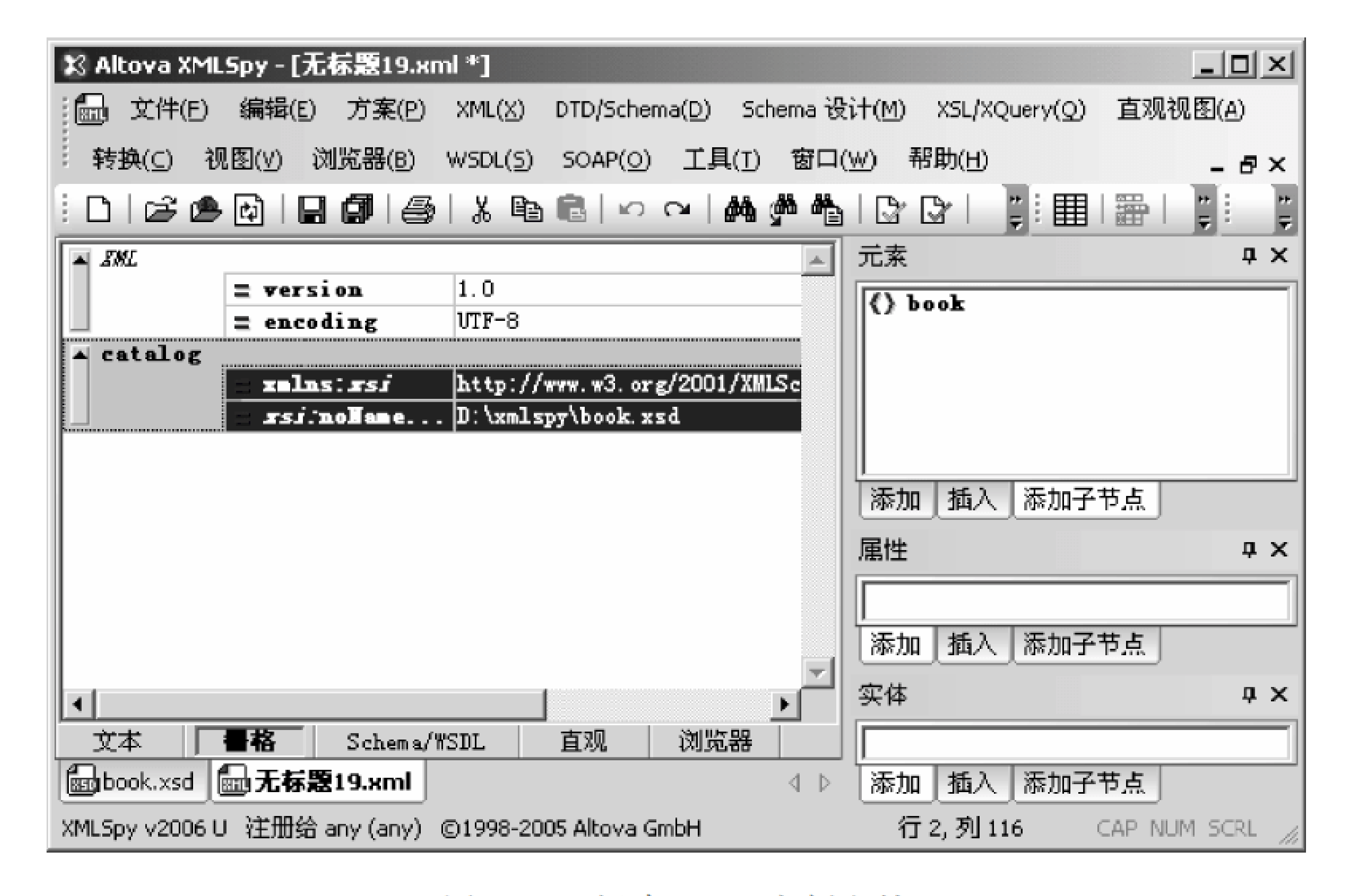



图 3-37　新建 XML 实例文档

在右侧的元素窗格中，单击"添加子结点"，显示 catalog 下包含的元素（根据 book. xsd 的定义），双击元素窗格的 book 项，添加一个 book 元素，如图 3-38 所示。

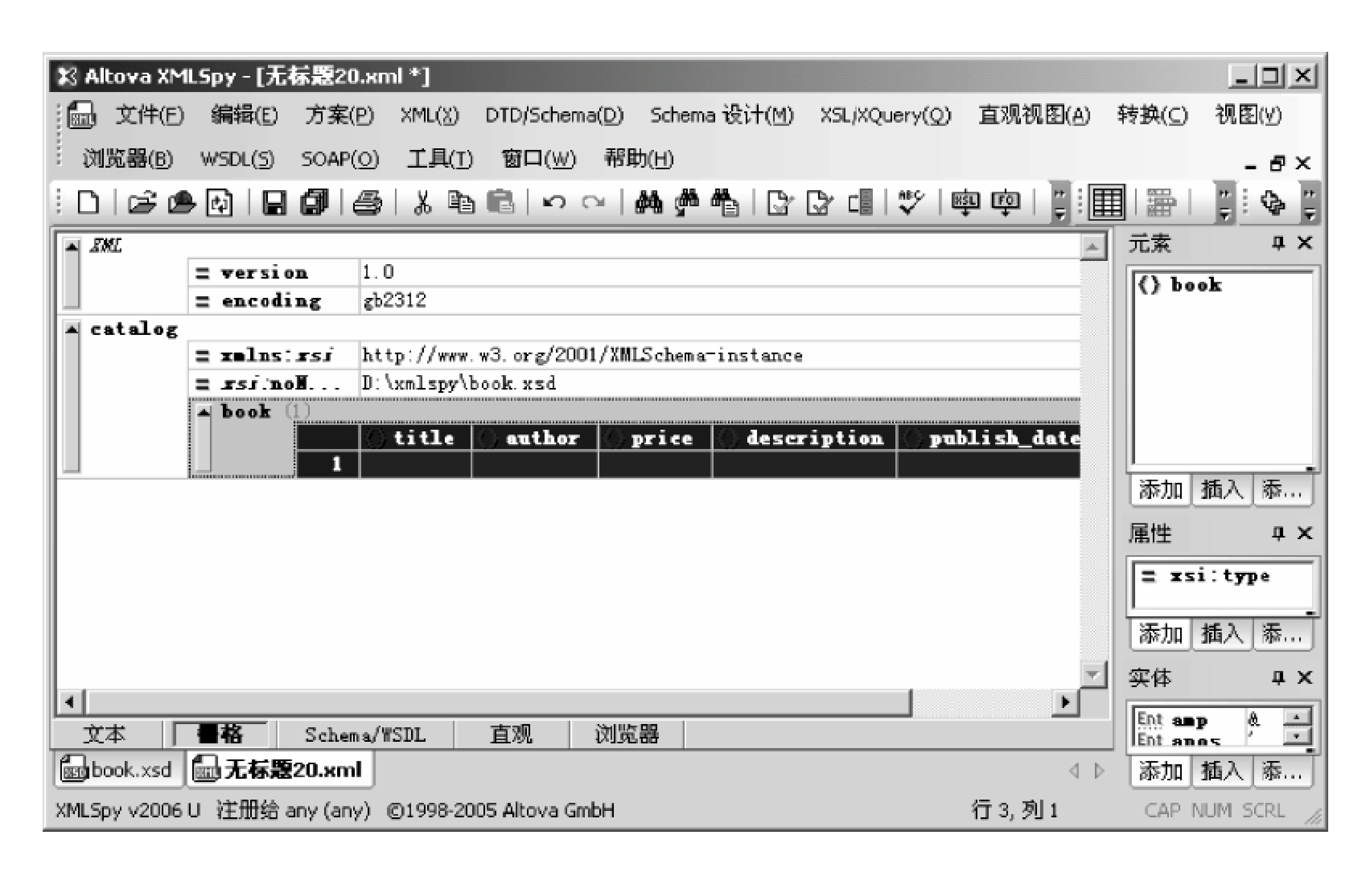



图 3-38 增加一个元素

单击“文本”,选择文本视图,如图 3-39 所示。

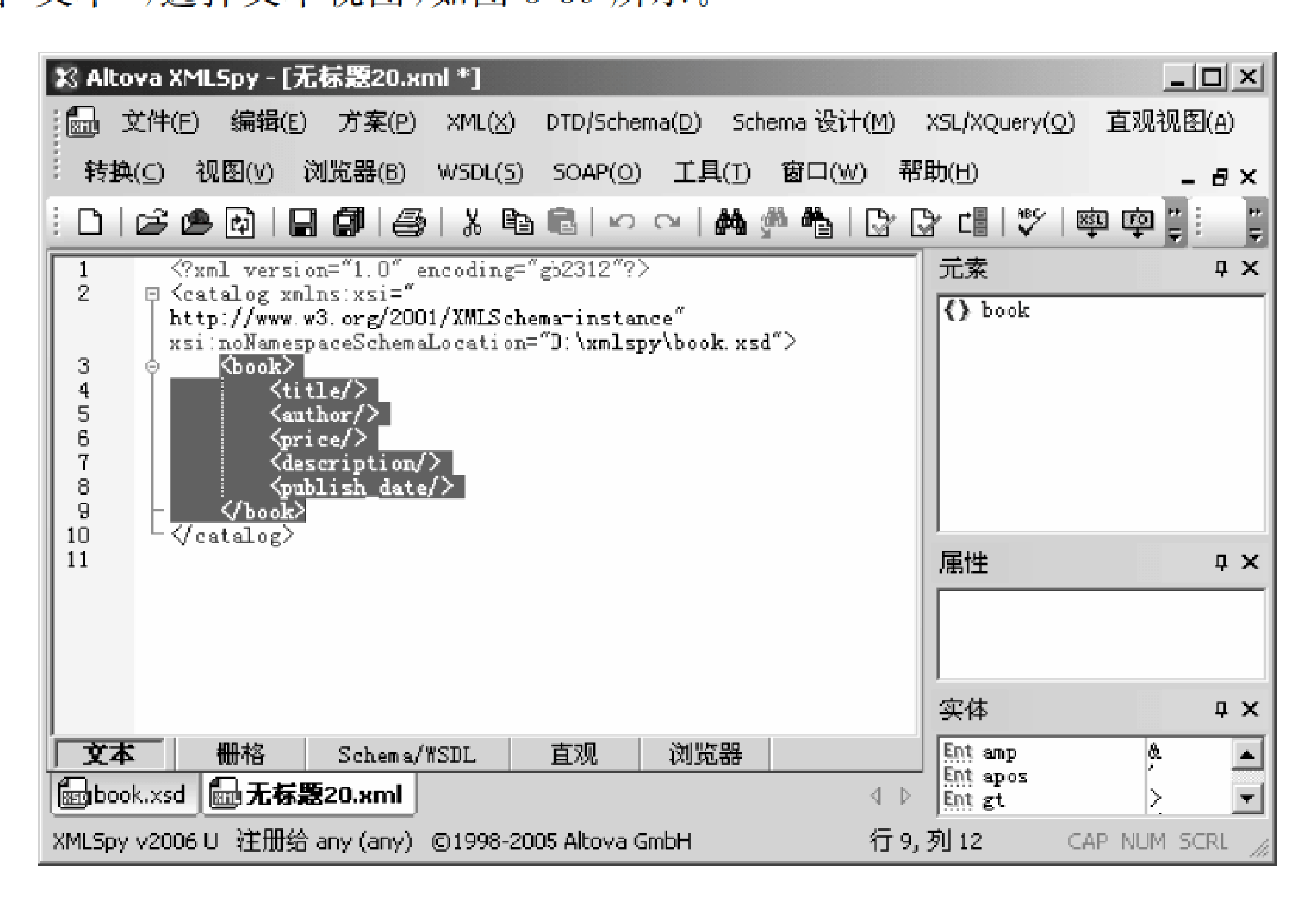



图 3-39 生成的 XML 文档代码

现在我们尝试修改 book 的子元素,代码如下:

```
<book id = "ISBNxxxxx">
    <title>Web Developing Technologies Guide</title>
    <author>Hao XW</author>
    <price>28.50</price>
    <description >An widely in - depth look and talk on creating applications on WEB</ description>
    <publish_date>2006 - 05 - 16</publish_date>
</book>
```

然后，单击“浏览器”按钮，如果文档检验通过，则显示如图 3-40 所示。

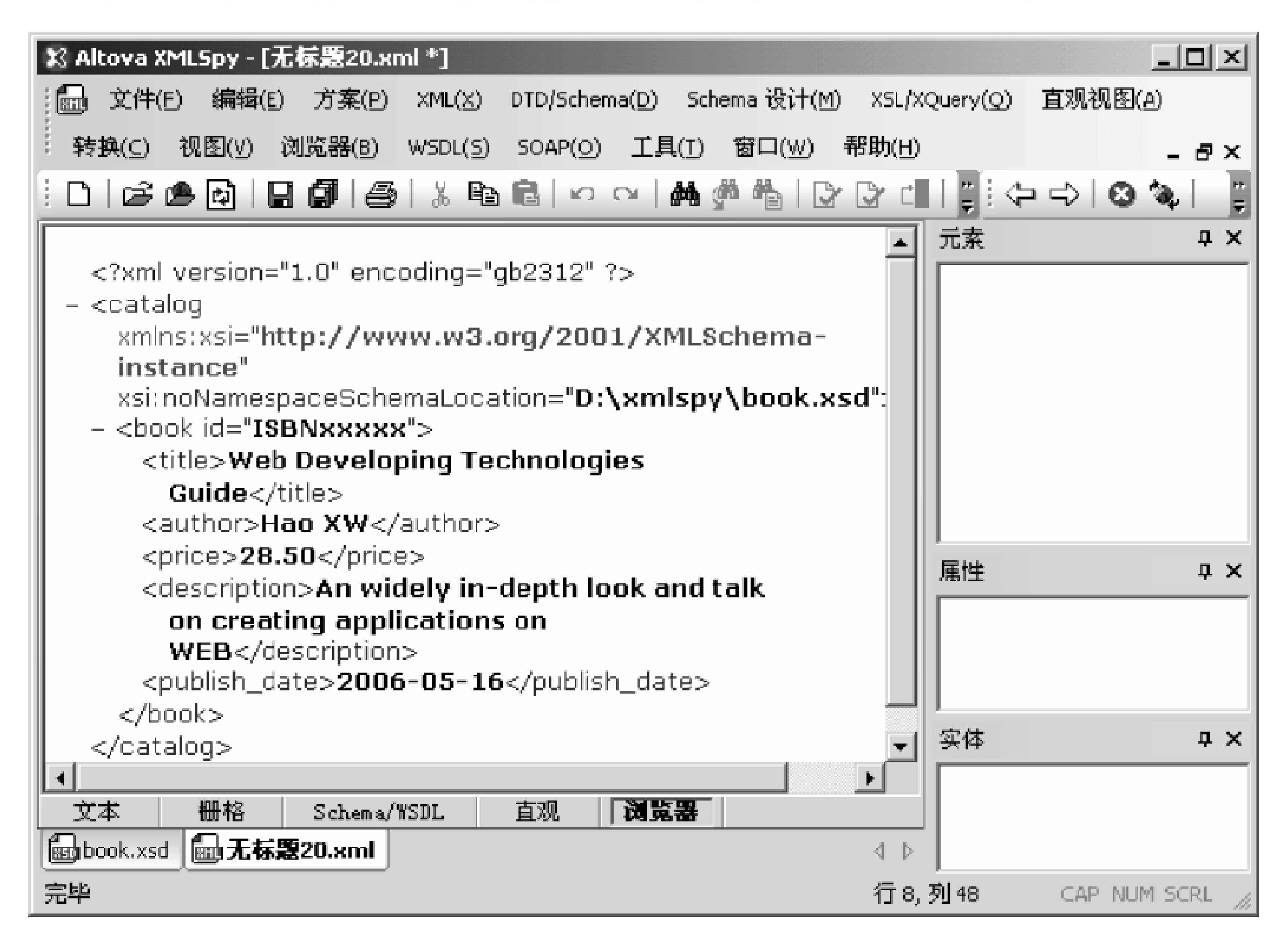



图 3-40　编辑后的 XML 文档

此时，回到文本视图，尝试修改<price>28.50</price>中的数据为一个字符串“xxx.xx”，然后单击“浏览器”按钮，如图 3-41 所示。

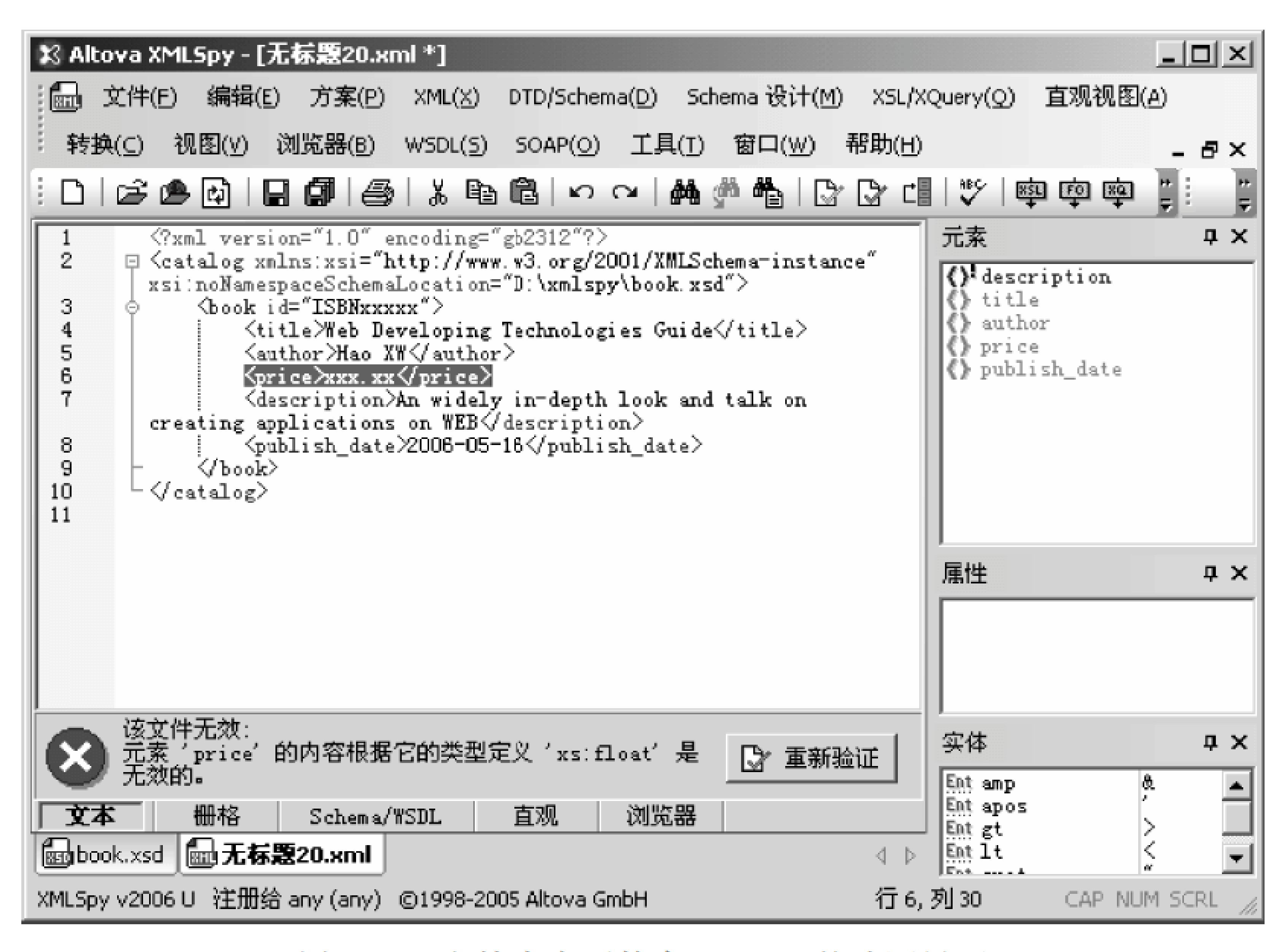



图 3-41　文档内容不符合 Schema 的验证界面

则系统检测到<price>标记的内容不是浮点数，即 XML 文档数据不符合 XML Schema 中的元素类型定义，数据无效。修改后，单击“重新验证”按钮，则会显示“该文件有

效”的提示信息。

## 3.5 其他相关技术

在 Web 中,除了标记语言 HTML、XML 以外,还有一些其他的技术,它们有着和 HTML 类似的名字,例如 DHTML、SHTML 等。但是,它们并不是 W3C 的标记语言规范,只是一些与 Web 开发相关的技术或者说开发模式。

### 3.5.1 DHTML 技术

DHTML 不是一种标准或规范,它只是一种将目前已有的网页技术、语言标准整合运用,制作出在下载后仍然能实时变换页面元素效果的网页的设计概念。DHTML 大致包含以下网页技术、标准或规范:

(1) HTML 4.0 规范。

(2) 客户端脚本语言 CSSL(Clent-Side Scripting Language),主要有 JavaScript(JS)、VBScript(VBS),大多数的浏览器都支持这些脚本语言。

(3) 文档对象模型 DOM(Document Object Model),DOM 是 W3C 推广的 Web 技术标准之一,它将网页中的内容抽象成对象,每个对象拥有各自的属性(Properties)、方法(Method)和事件(Events),可以通过上面的客户端脚本语言编程控制。

(4) 层叠样式表单 CSS 技术,CSS 是 HTML 的辅助设计规范,用来弥补 HTML 在排版上所受的限制导致的不足,它是 DOM 的一部分。理论上说,通过 CSSL 动态地改变 CSS 属性可以做出丰富的页面视觉效果。

总之,DHTML 技术,就是以 HTML 为基础,运用 DOM 将页面元素对象化,利用 CSSL 控制这些对象的 CSS 属性以达到网页的动态视觉效果。

### 3.5.2 SHTML 技术

SHTML 不是一种 HTML 规范,而是一种网站设计、开发与维护技术,可以说成是一种 Web 服务器 API,其指令可以在 Web 服务端运行,以产生动态的 HTML,是一种类似于 ASP 的基于服务器的网页制作技术。

SHTML 文件包含服务器端包含指令(Server Side Include,SSI),具有特殊的 SSI 文件扩展名,默认扩展名是.stm、.shtm 和.shtml。在下载网页时,具有 SSI 功能的 Web Server(例如基于 UNIX 平台的 Web 服务器 Netscape Enterprise Server 等均支持 SSI 命令)首先扫描 SHTML 文件,解释执行其中的 SSI 指令,然后跟一般的 HTML 一起发送到客户端。

服务器端包含类似于程序设计中的常量定义,可以使系统维护更加方便。例如,将内容发送到浏览器之前,可以使用 SSI 包含时间/日期戳、版权声明或供客户填写并返回的表单。对于在多个文件中重复出现的文本或图形,使用包含文件是一种简便的方法。将内容存入一个包含文件中即可,而不必将内容输入所有文件,对内容的所有更改只需在一个地方即可完成。

服务器端包含可以使网站维护更简单，可以定义一些相对稳定的页面模板，通过服务端包含指令，引入具体的内容，自动生成网页，而不需修改网站目录结构。使用服务器端包含，可以使页面更新容易，例如批量更新 banner、版权等信息，从而提高网站的维护效率。

## 思　考　题

1. 什么是 HTML？简述 HTML 文档的基本结构。

2. HTML 文档可分成文档内容和标记两个部分，关于文档标记，回答下列问题：

(1) 标记一个图片的标记是什么？该标记有哪些常用的属性？说明每个属性的含义。

(2) 标记一个超链接的标记是什么？该标记都有哪些属性？说明每个属性的含义。

3. 在 HTML 中，如果要改变标记的默认显示样式，有哪些方法？

4. 在 HTML 中，使用<input type="file">上传文件，如何设置文件在服务器上的存储路径？

5. 什么是层叠样式表 CSS？使用 CSS 有什么好处？

6. XML 与 HTML 相比有什么本质不同？

7. 一个 XML 文档有哪几个组成部分？简述每一部分的功能。

8. 在 XML 文档中，文档类型定义 DTD 和文档架构的目的是什么？

9. 编写关于客户资料的 Schema，用 XMLSpy 检验编写的 XML 文件的有效性。

10. 对于 XML 中的名称空间，回答下列问题：

(1) 什么是 XML 名称空间？

(2) 在 XSL、Xpath、Xlink 等语言规范中，都分别指定了一个名称空间超链接，是否意味着名称空间都对应一个网址？要使用名称空间，必须要在线连接吗？

11. XML 技术实现了数据和显示的分离，关于 XML 文档内容的显示，回答下列问题：

(1) 如果不声明 XML 文档的显示样式，在浏览器中加载一个 XML 文档时，显示的结果是什么？

(2) XML 文档树显示了文档数据的结构，如何进行 XML 文档的格式化显示？

12. 在 XML 技术中，相关的规范和标准较多，回答下列问题：

(1) 什么是 XSL？XSL 和 XML 是一种什么样的关系？

(2) XPath、Xquery、Xlink 以及 Xpointer 等规范分别是什么意思？它们和 XML、XSL 有什么关系？

13. 有一个网上购物站点，针对图书类商品，包含下列信息：封面图片、书名、作者、出版社、出版日期、价格，要求：

(1) 编写一个 XML 文档，来描述图书类商品，要求对于“封面图片”应定义一个图片超链接元素。

(2) 编写对应的 XSL 文件，完成图书 XML 文档的显示。

14. 关于 XML 文档对象模型 DOM，回答下列问题：

(1) Document 对象有哪些常用的属性和方法？

(2) Node 对象有哪些常用的属性和方法？

15. Altova XMLSpy 是一个什么样的软件？具有哪些典型功能？

# 第 4 章

# 网页设计与制作

在互联网中，网页是用户和 Web 的主要界面，用户通过网页来进行信息浏览和实现与 Web 应用的交互。从本质上讲，网页是存储在 Web 服务器上的一个个 HTML、JSP、ASP、PHP 等各种类型的文档在浏览器中的显示，一个 Web 应用或者一个网站是由大量的网页构成的。

和传统的软件系统相比，Web 应用（网站）不是基于菜单命令，而是通过页面间的超链接表达和实现业务逻辑。相对于传统软件中的菜单命令，Web 中页面之间的超链接设计更加灵活，页面不需要编译，这增加了 Web 应用设计、开发和运行的灵活性。此外，网页作为 Web 应用的用户界面，更多地强调了它的艺术性效果，追求用户体验，这使得 Web 应用的设计融入了更多非技术的要素，网页设计成为 Web 设计的主要内容。

## 4.1 网页设计基础

和传统的程序界面不同，网页是内容和艺术的综合体。网页在实现 Web 应用系统功能的同时，表现出更大的灵活性、随意性和艺术性。Web 开发团队的人员也更加多样，除了传统的系统设计人员、代码开发人员外，开始出现了美工人员，Web 设计已经越来越明显地分成面向业务逻辑的功能性设计（交互设计）和面向用户体验的视觉设计两个不同的层面。

### 4.1.1 页面功能与内容设计

在 Web 应用或 Web 站点中，所有的信息和业务逻辑都通过网页来实现。页面作为用户和 Web 的界面，页面的功能首先是一种交互功能。因为一个 Web 站点通常由大量的页面文件构成，因此对页面的功能划分、存储和组织应按照软件系统分析、设计和开发相关的方法和模式进行，包括生命周期法、原型法和 MVC 设计模式等。

综合利用生命周期法、原型法和 MVC 设计模式，对整个 Web 站点进行系统分析和功能设计，然后规划文件夹结构、数据库以及页面。和传统的软件系统开发相比，Web 站点中的一个页面，类似于传统软件系统中的一个函数文件，只是页面之间的调用是通过超链接，或者页面中包含的表单的 action 属性完成的，而不是传统 C 中的函数调用。

在 Web 开发中，模型—视图—控制器（Model-View-Controller，MVC）设计模式是一种

比较流行的设计模式。MVC是Xerox PARC在20世纪80年代为编程语言Smalltalk－80发明的一种软件设计模式，其核心思想是在系统开发中把商业逻辑、界面显示和数据进行分离，强制性地使应用程序的输入、处理和输出分开，分成相对独立而又能协同工作的3个组成部分：模型（Model）、视图（View）、控制器（Controller），它们各自处理自己的任务。

模型表示企业数据和业务规则，在MVC的三个部件中，模型拥有最多的处理任务，实现具体的业务逻辑、状态管理的功能。被模型返回的数据是中立的，就是说模型与数据格式无关，这样一个模型能为多个视图提供数据。由于应用于模型的代码只需写一次就可以被多个视图重用，所以减少了代码的重复性。

视图是用户看到并与之交互的界面，通常实现数据的输入和输出功能。

控制器控制整个业务流程，实现View层跟Model层的协同工作。控制器接受用户的输入并调用模型和视图去完成用户的需求。当单击Web页面中的超链接和发送HTML表单时，控制器本身不输出任何东西和做任何处理。它只是接收请求并决定调用哪个模型构件去处理请求，然后确定用哪个视图来显示模型处理返回的数据。

根据MVC设计模式，可以将网页功能进行分类，分为：（1）用于输入输出的页面（视图）；（2）服务端脚本程序页面（模型），这类页面不在浏览器中显示，主要是负责数据的查询、存储等；（3）导航页面，类似MVC中的控制器，也类似于传统程序中的菜单，实现页面之间的调用和导航，典型的导航页面就是站点首页。将网页按照MVC设计模式进行分类可以更好地规划一个Web站点，保证站点的可扩展性、灵活性和可维护性，这也是所有的软件设计模式所追求的目标。

页面的功能划分主要服务于Web应用的交互，提高网站的易用性，协助用户顺利地完成期望的任务流程。对于视图类和导航类页面，还必须考虑其用户体验，网页的内容设计也丰富多彩，包括网站标志、导航、菜单、图片按钮、表单样式、表格数据文字表现、新闻、公告、讨论区、blogs、友情链接、广告条、版权信息等。对网页内容的表现形式，应根据内容和用户特点，选用文本、图片、动画等不同的媒体形式来展示，以产生更好的用户体验。

### 4.1.2　页面布局设计

在Web应用中，网页是用户和Web的人机界面。在Web设计中，网页布局越来越重要，只注重内容，而忽视页面布局的网页很难能产生较好的用户体验。实际情况是，用户对页面的体验第一印象就是页面的栏目和布局，然后才是页面内容。只有当网页内容和网页布局完美地融合时，才能产生最好的用户体验。

#### 1. 网页布局设计方法

新建页面就像一张白纸，没有任何表格、框架和约定俗成的东西。接下来，设计人员根据页面内容、浏览用户等因素对页面布局进行设计。进行页面布局，通常需要先在纸上或者利用Photoshop等画图程序来设计草稿，画出页面布局的草图，然后再深入加工，最后定稿。在进行页面布局设计时，需要考虑以下几个因素：

（1）页面尺寸。页面尺寸和显示器大小及分辨率有关系，同时浏览器本身（如菜单栏、工具栏等）也将占去一定的屏幕空间，这在页面布局设计时不需考虑。一般情况下，如果屏

幕分辨率为800×600,则页面的显示尺寸一般为780×428像素,如果分辨率为1024×768,页面显示尺寸为1007×600。此外,如果页面尺寸采用像素值,可以在<body></body>标记对内增加<center></center>标记对,将<body></body>内的所有页面内容居中,以应对不同的屏幕分辨率。

(2) 整体造型。是指页面的整体形象,虽然显示器和浏览器都是矩形,但对于页面的造型,可以充分运用自然界中的各种形状以及它们的组合,例如矩形、圆形、三角形以及不规则边界的图形等。

不同的形状所代表的意义是不同的。例如:矩形代表着正式、规则,大多数政府网页都是以矩形为整体造型;圆形代表着柔和、团结、温暖、安全等,许多时尚站点喜欢以圆形为页面整体造型;三角形代表着力量、权威等,许多大型的商业站点为显示它的权威性常以三角形为页面整体造型;大部分的游戏场景则使用了不规则的图形。

(3) 页头。页头又称为页眉,页头常放置站点的名字、公司标志或旗帜广告。页头的内容和设计风格直接影响到整个页面的协调性。对于站点首页,为了有效利用屏幕空间,许多网站的页头中,除了放置公司标志,往往在页头右侧还放置了一组超链接。

在页头下方,往往是一个Flash动画,将页头和下面的内容分开,从而产生较好的视觉效果。大多数的门户网站首页都采用了上述的页头设计,例如Sina、163的首页页头。

(4) 页脚。页脚和页头相呼应,页头放置站点主题,页脚则放置制作者或者公司信息。

(5) 菜单。和传统的程序类似,在网页上也通常组织菜单,菜单其实都是超链接,将这些超链接组织成树形目录结构或弹出式菜单形式。

(6) 超链接。在具有导航功能的页面中(相当于MVC中的控制器),包含了大量的超链接,以链接到其他页面。可以按照链接的内容,将超链接组织成不同的超链接区。例如,许多门户网站的各种版块栏目,就是一个个的超链接区。超链接可以单独出现,也可以组织成多行多列的超链接区,也可以组织成横向菜单条,或者纵向的超链接区。

#### 2. 常见的网页布局

在数以亿计的Web页面中,网页的布局可谓千差万别。由于网页的功能不同,表达的内容不同,人们的审美情趣不同,因此,我们不可能要求设计人员设计统一的网页布局。虽然如此,有许多的页面布局是很多设计人员喜欢使用的,这些常见的页面布局是:

(1) 基于栏目的页面布局。对于内容很多的网页,通常将页面按照栏目进行组织,组织成多个矩形区域(内容板块),每个区域包含一组超链接,形成一个超链接区。超链接区通常包含一个栏目标题,栏目之间通过色彩块来区分。为了增加视觉效果,在栏目之间,或矩形块栏目内部,通常插入一些Flash动画广告。

基于栏目的页面布局主要用于导航页面设计,大部分的门户网站首页即采用基于栏目的页面布局形式,例如,Sina、Yahoo、163、265上网导航等。网页布局示例如图4-1所示。

基于栏目的页面布局主要应用于一些商业网站的首页。为吸引用户,增加用户访问量,从商业运营的目的出发,网站的内容很多,分成了不同的超链接区域(板块)。为节省屏幕空间,在栏目内往往还使用标签,以组织更多的内容。

(2) 整幅效果型布局。页面采用大幅图片或Flash动画,在底部加一"登录"按钮,通常用于站点首页。特点是页面美观,可以较好地展示企业形象,缺点是登录速度较慢。




图 4-1　基于栏目的页面布局示例

(3)“口”型页面布局。页面一般上下各有一个广告条,左面是主菜单,右面放友情链接等,中间是主要内容,这种页面布局也经常用于一些站点的首页设计。

“口”型页面布局的优点是充分利用版面,信息量大。缺点是页面拥挤,不够灵活。也有将四边空出,只用中间的窗口型设计。

(4)“工”型结构布局。所谓“工”型结构布局,是指将页面分成上、中、下三个部分,页面顶部为横条网站标志+广告条,下面是版权等信息,中间为主要内容,又分为左右两个部分,左面为主菜单,右面显示内容的页面布局形式。示例如图 4-2 所示。

“工”型页面布局是网页设计中用的最广泛的一种布局方式,这种布局的优点是页面结构清晰,主次分明。缺点是规矩呆板,如果细节和色彩上搭配不好,很难给人留下印象,不适宜做前卫的和个性化强的站点。

(5)自顶向下层次结构布局。将页面自顶向下分成几个平行的区域,顶部是页头,接下来的区域分别放置超链接块,最下面的区域显示具体的文章正文内容。一些文章页面或注册页面等经常采用这种类型的页面布局。

(6)自由式结构布局。上述的结构布局可称作“传统型”页面布局,还有一些页面结构布局打破了传统的页头、页尾、菜单、栏目、超链接区域等布局模式,把页面设计成一张极具创意的广告作品。这种页面的结构布局通常用精美的图片、网站标识性图案(Logo)或变形的艺术化文字作为设计中心进行主体构图。菜单、栏目条等则按次要元素处理,自由地安排在页面上,起到点缀、修饰、均衡页面的效果。

自由式结构布局的优点是页面靓丽、现代、轻松、节奏明快,很容易让访问者驻足欣赏。缺点是下载速度缓慢,文字信息量少,信息的逻辑表达能力弱,浏览者不易直奔主题,信息查

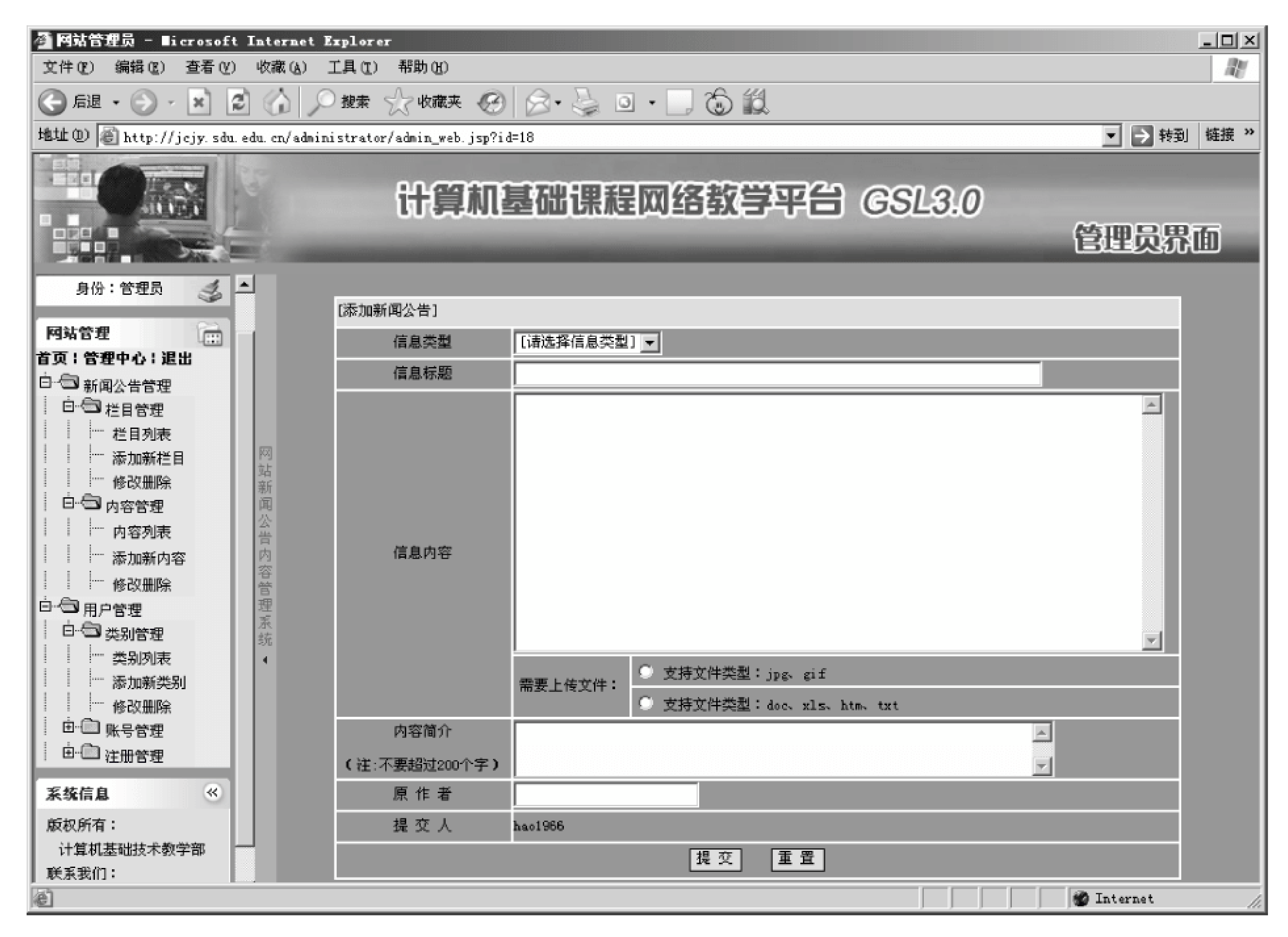



图 4-2 “工”型页面布局示例

找麻烦。自由式结构布局一般用在时尚类站点上，如时装、化妆品等以崇尚现代、美感为主题的站点，专业性的商务站点不宜采用。

页面布局设计是一项复杂的创意工作，在进行页面布局设计时，需要根据站点的性质（例如专业性商务站点、前卫现代时尚类站点等）、页面的功能、是否首页、页面内容的多少进行精心策划，才能设计出功能性和视觉效果好的页面。

### 4.1.3 页面视觉设计

视觉设计是指利用视觉符号来传递各种信息的设计，其应用的范畴很广，可以包括工业产品设计、广告设计、新媒体设计、服饰设计等等。在 Web 中，则有网页的视觉设计。在网页中，视觉设计和功能性设计不同，它没有功能性要求，没有太多的理论约束，更多的是体现出感性和个性元素，服务于人的审美情趣。我们可以把页面视觉设计分成图形、字体和色彩几个方面。

在页面的视觉效果的诸多影响因素中，颜色设计产生的视觉效果最直接和明显，不同功能的网站，其颜色的主色调设计也不相同。大部分网站都追求一种明快的颜色设计，例如，绿色、深蓝、橙色和粉红色，这些颜色都十分人气。其次是颜色的搭配，浓重的主色调表明了幽默的态度，也有助于人们迅速注意到页面上的重要元素。

关于图形的应用，现在许多 Web 2.0 站点的页面都避免使用照片，大都选择简洁的图标和截图。可以将图形的应用分为几个方面：(1)用于信息反馈。信息反馈一般有以下五种情况：成功、失败、询问、警告、错误/异常，视觉辅助图必须表达每种情况的准确含义。(2)增加趣味性。为了增加趣味性，通常用图形来表现一种状态，例如，表现喜怒哀乐的简单

图形。当使用图形时，图形的设计一定要注意它的准确性，否则会起到相反的效果。例如，用一个惊讶的表情来表示警告，但往往被误以为是询问或者出现了异常。

对于文字的字体，更多的应选择大号字。另外，要注意发挥空白的作用，巧妙利用空白可以提高页面的易读性和易用性。空白可以分离出重要信息，使眼睛得到休息，并给人以冷静和有秩序的感觉。

此外，从总体上来讲，页面设计还需要遵循几点一般性的原则，包括：(1)平衡，避免一边倒、头重脚轻，可以是均衡对称，也可以是不均衡对称；(2)重点突出，有且只有一个视觉趣味中心(Center of Visual Interest，CVI)，可以通过对位置、对比和比例的处理来实现；(3)简化原则，去除或尽量弱化干扰，突出一点；(4)重复原则，通过合理重复，形成一个视觉模式；(5)统一一致，保持风格的一致性；(6)便利性、可读性与易辨认，便于读者阅读。

最后简单地总结一下，功能性设计通常是一种交互设计，其主要目标是提高系统的易用性；视觉设计的主要目标不是功能性的，它的目标是提高人们的视觉感受。视觉设计是一项复杂的艺术创造，需要很深厚的艺术文化积淀。虽然人们不断推崇务实的简约设计概念，但是另类的创新艺术表现也不时地出现在各种时尚和前卫的网站中。

### 4.1.4　页面效果设计

当网页的整体布局确定后，接下来就是效果设计了。效果设计就是利用 Photoshop 等图形图像处理工具，按照页面的布局设计，来设计页面的完整图片。然后对图片进行切图，为下面的页面 HTML 代码编写准备 images。图 4-3 是一个 Blog 页面的设计效果图。




图 4-3　页面设计效果图示例

当页面设计效果图完成后,接下来就是进行切图。将效果图中的元素剪切为一个个小的图片,以保证网页的浏览速度。首先要分析页面效果图,它是未来网页的显示效果。对于上述的页面效果图,切图时应从以下几个方面考虑:

(1) 在页面上部是一行带有背景的菜单,因此,我们可以从中切一个像素宽度为 1 的小图片,在页面 HTML 代码中,利用该图片实现 Table 单元格中的背景横向填充,以达到效果图的显示效果。如果需要纵向填充,可以取高度为 1 的小图片,进行纵向填充。

(2) 对于页面主体中的一些栏目标题,可以直接切为小图片,这些图片的文件很小。

其他的页面元素,操作类似。最后将这些切图图片存储到一个特定的 images 文件夹中,它是下一步进行 HTML 页面设计的素材。

在页面效果设计中,还涉及了其中的文字,对于文字内容,可以通过 CSS 样式表技术来定义文字的样式。如果布局采用 Table,表格属性要定义 CSS,从而实现页面布局和显示样式的分离,增加页面维护的灵活性。关于 CSS 的使用,请参考第 3.2.9 小节的内容。

## 4.2 使用 FrontPage

FrontPage 是微软开发的一种可视化的网页制作和站点管理工具。FrontPage 的功能可以分为站点管理和网页制作两大方面。FrontPage 有 FrontPage 2000 和 FrontPage 2003 两个常用版本,两者没有本质的区别,下面我们将以 FrontPage 2003 为例,重点介绍网页制作工具的网页制作方法。

### 4.2.1 FrontPage 主窗口

运行 FrontPage 2003,FrontPage 打开后,首先需要打开一个网站(即打开站点对应的主目录文件夹),一个站点不一定是一个 Web 服务器上真正运行的网站,可以是本地机上的一个文件夹。接下来可以在该文件夹下新建网页,或对网页进行维护,主界面如图 4-4 所示。

在 FrontPage 2003 主窗口中,可以显示网站文件夹列表,双击某个网页文件,则在右边打开该文件。在网页文件设计窗口的左下角,有一组网页视图模式的选项标签,分别是“设计”、“拆分”、“代码”和“预览”四种显示模式。如果是框架网页,还含有其他的几个标签。单击标签,可以选择不同的工作模式。

#### 1. 设计模式

这是一种可视化的网页设计模式。在设计模式下,用户可以采用“所见即所得”的方式设计、编辑和修改网页,系统将自动生成对应的 HTML 代码。代码可在“代码模式”或“拆分模式”下显示。

#### 2. 拆分模式

选择拆分模式,则客户工作区分为上下两个区域,下面的区域显示设计的网页,上面部分显示对应的 HTML 代码。因此,该视图能够非常好地将设计中的网页元素和对应的 HTML 代码联系起来,并且便于手工调整。

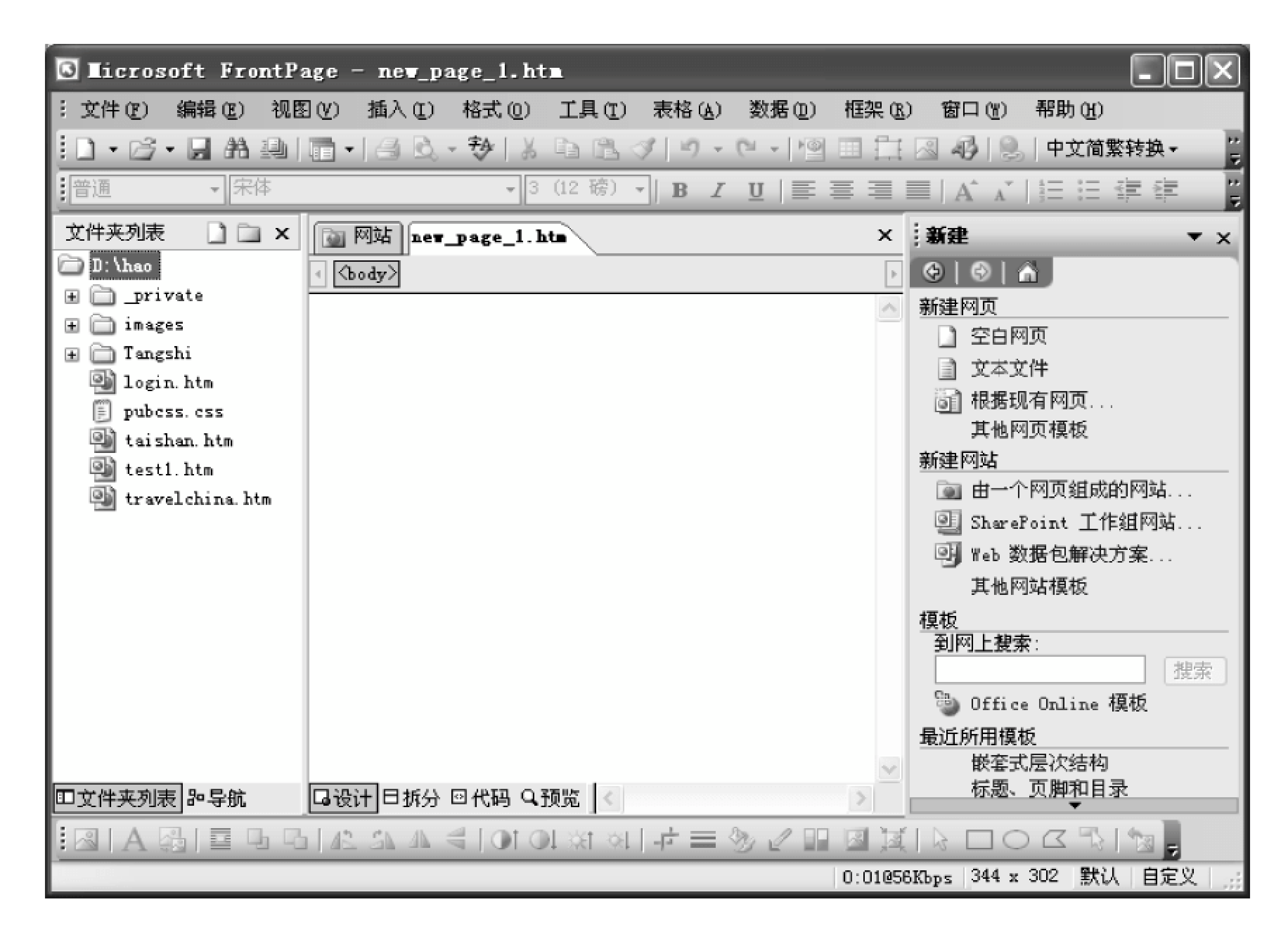



图 4-4　FrontPage 2003 主窗口

### 3. 代码模式

在该模式下，将显示设计中的网页对应的 HTML 代码。一般情况下，用户在设计模式下设计网页，然后在代码模式下查看网页对应的 HTML 代码。另外，用户也可以直接在该模式下编辑、修改其中的 HTML 代码，适用于那些对 HTML 比较熟悉的用户。如果网页中包含一些脚本程序，则这些脚本程序也需要通过代码模式进行编辑。

如果用户需要复制其他网页的 HTML 代码，也不许选择代码模式，将被复制的内容复制到当前网页的适当位置，而不应该在设计模式下进行复制。

### 4. 预览模式

选择预览模式，将对设计的网页进行预览，与通过浏览器所看到的网页一致。另外，如果网页中包含脚本程序，在程序的调试阶段通过 FrontPage 的预览模式可以很容易查出程序出错的位置和错误性质。

## 4.2.2　网站的新建与维护

FrontPage 不仅是一个页面制作工具，同时，它还可以进行站点管理操作，包括创建、删除、打开、发布站点，以及进行站点的维护等工作，也可以对站点的文件和文件夹进行操作以及对站点进行安全性管理。

### 1. 新建网站

创建一个 Web 站点,本质上就是建立一个主目录,然后在主目录下创建子目录和网页文件。用户可以手工地建立站点目录结构,通过资源管理器可以定位具体的文件夹和文件。为了管理方便,现在的网页制作工具都提供了新建站点和站点管理,以方便对站点的维护。

在 FrontPage 中,提供了创建站点的模板和向导,通过它们可以很容易地建立起一个新站点,即创建站点目录结构。操作方法是:执行“文件”菜单中的“新建…”命令,显示“新建”任务窗格,在“新建网站”区域,单击其中的任何一个超链接,都可以打开“网站模板”对话框,如图 4-5 所示。




图 4-5 “网站模板”对话框

单击“个人站点”,在“指定新网站的位置”文本框中,输入一个文件夹名,该文件夹即为站点主目录,单击“确定”按钮。向导则自动创建一个站点主目录,并根据模板,创建相应的子文件夹和网页文件,如图 4-6 所示。

在屏幕的左边,显示了网站的目录结构,这便于站点的管理和维护。如果不显示文件夹列表,此时,在“视图”菜单中,执行“文件夹列表”命令,即可打开站点文件夹列表,以便于对站点的操作。

在网站窗口的下面,显示“文件夹”、“远程网站”、“报表”、“导航”、“超链接”和“任务”多个选项卡,单击“超链接”选项卡,可以在网站中显示各个网页之间的超链接,如图 4-7 所示。

超链接视图可以清晰地显示页面之间的连接关系,对于站点的维护有很大的帮助,容易对站点逻辑建立全面的概念。

### 2. 本地网站维护

一个站点,本质上就是由一个文件夹以及其中包含的所有网页文件构成的。如果在 Web 服务器上,一个站点是真实运行的网站。如果是在一台桌面机上,例如 Windows XP 上,我们同样可以创建一个站点,如果仅仅是纯 HTML 文件,不包含服务器脚本,它同样可以运行。此外,对于任意一个文件夹,用户也可以作为一个站点来打开,在其中新建网页。




图 4-6　新建个人站点

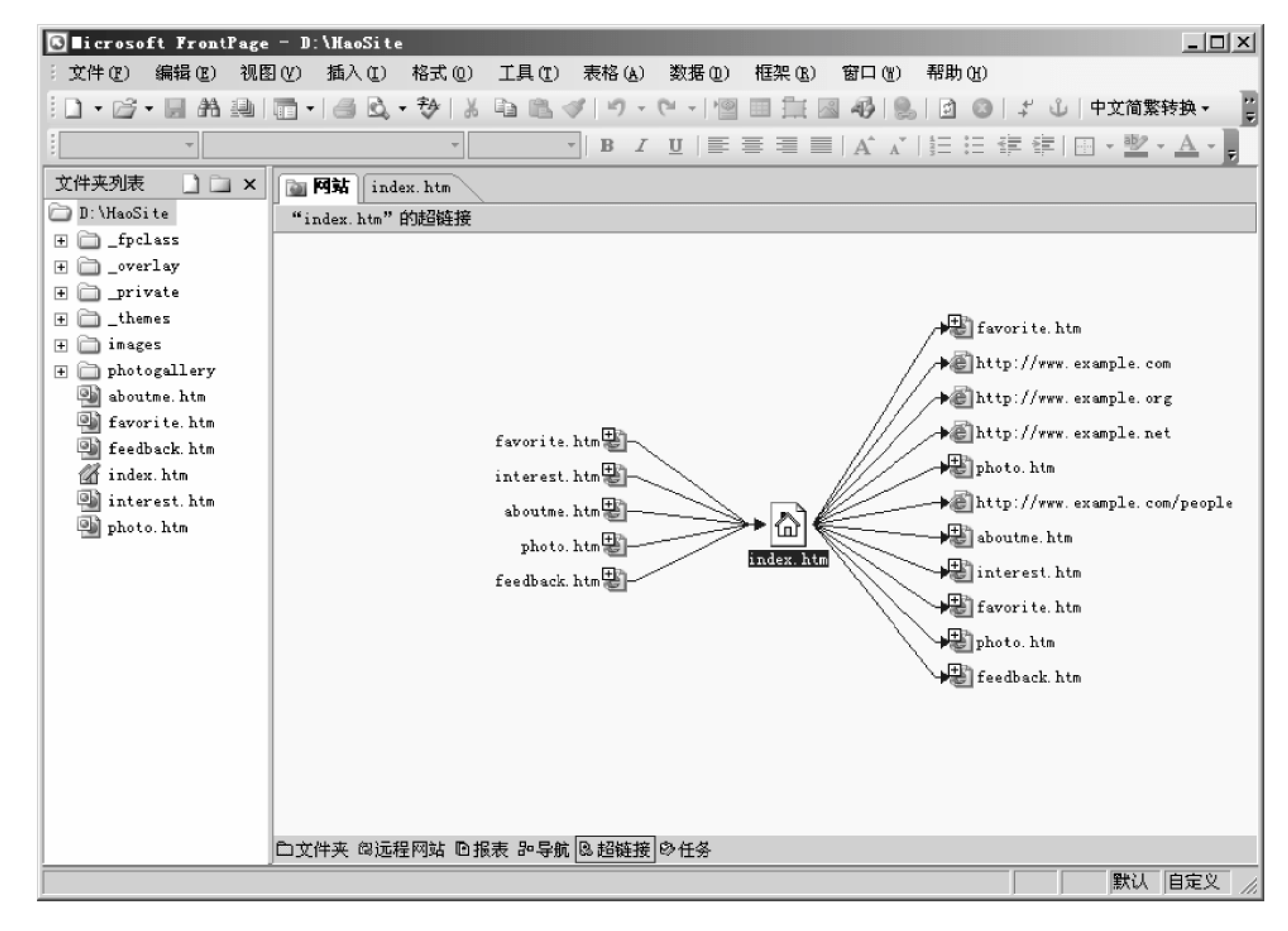



图 4-7　网站超链接视图

因此,无论是在Web服务器上,还是在桌面机上,都可以使用FrontPage对一个本地的文件夹(看做网站主目录)进行维护,具体步骤如下:

(1) 在FrontPage中,执行"文件"菜单中的"打开网站…"命令,打开"打开网站"对话框,如图4-8所示。




图4-8 "打开网站"对话框

在查找范围中,选择网站对应的主文件夹,网站名称后面的文本框中不需要输入内容,单击"打开"按钮,则文件夹名即为网站名,然后显示网站视图,显示网站目录结构。

如果打开的文件夹是一个普通的文件夹,FrontPage将在该文件夹下添加两个子文件夹"_private"和"images"。

(2) 执行"视图"菜单中的"文件夹列表"命令,显示网站文件夹结构。接下来就可以对站点进行各种各样的维护和管理操作了。

站点的维护主要是指在站点主目录下新建、删除、修改文件夹。在某个文件夹中,新建网页、修改网页或删除网页文件等。在网站"文件夹列表"中,在某个文件夹上右击鼠标,打开快捷菜单,可以执行"删除"、"重命名"、"新建(文件夹、空白网页)"等操作。

如果在某个网页文件上双击,则打开该网页文件,显示网页设计视图,可以进行网页的制作和内容修改。

### 3. 远程网站维护

使用FrontPage,可以对Windows平台下的IIS网站进行远程维护。所谓远程站点维护就是利用FrontPage在本地计算机上打开一个文件夹,作为本地网站;然后可以在本地网站编辑网页文件,然后发布到远程网站,或者将远程网站的文件下载到本地,以及在本地网站和远程网站之间进行网页文件同步。

要对一个网站进行远程维护,需要对网站本身进行相应的设置,具体设置内容如下:

(1) 在安装的IIS组件中,选择安装"FrontPage 2002 Server Extensions"。

(2) 在安装的IIS组件中,选择安装"万维网服务"组件,选择其中的"WebDAV发布"等组件,如图4-9所示。

(3) 在IIS控制台中,启用Web服务扩展中的相应服务,如图4-10所示。

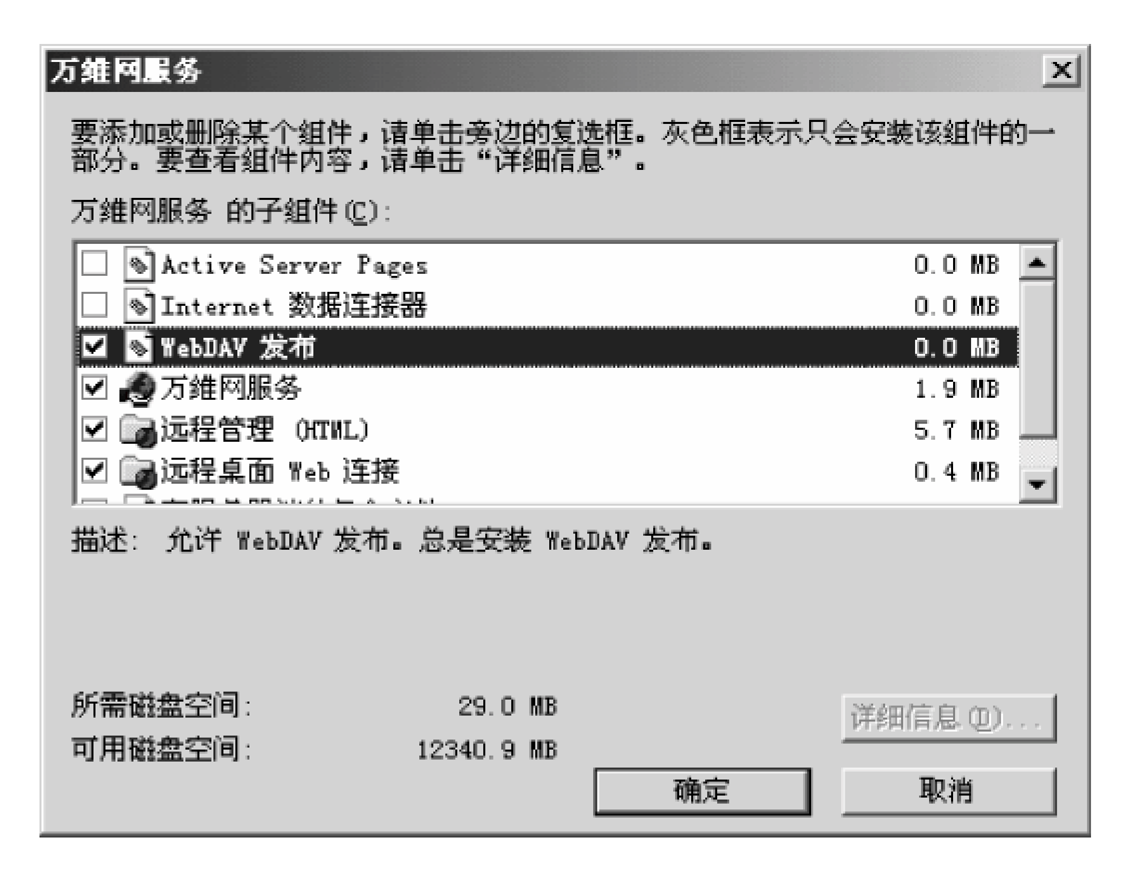



图 4-9　选择安装万维网服务组件

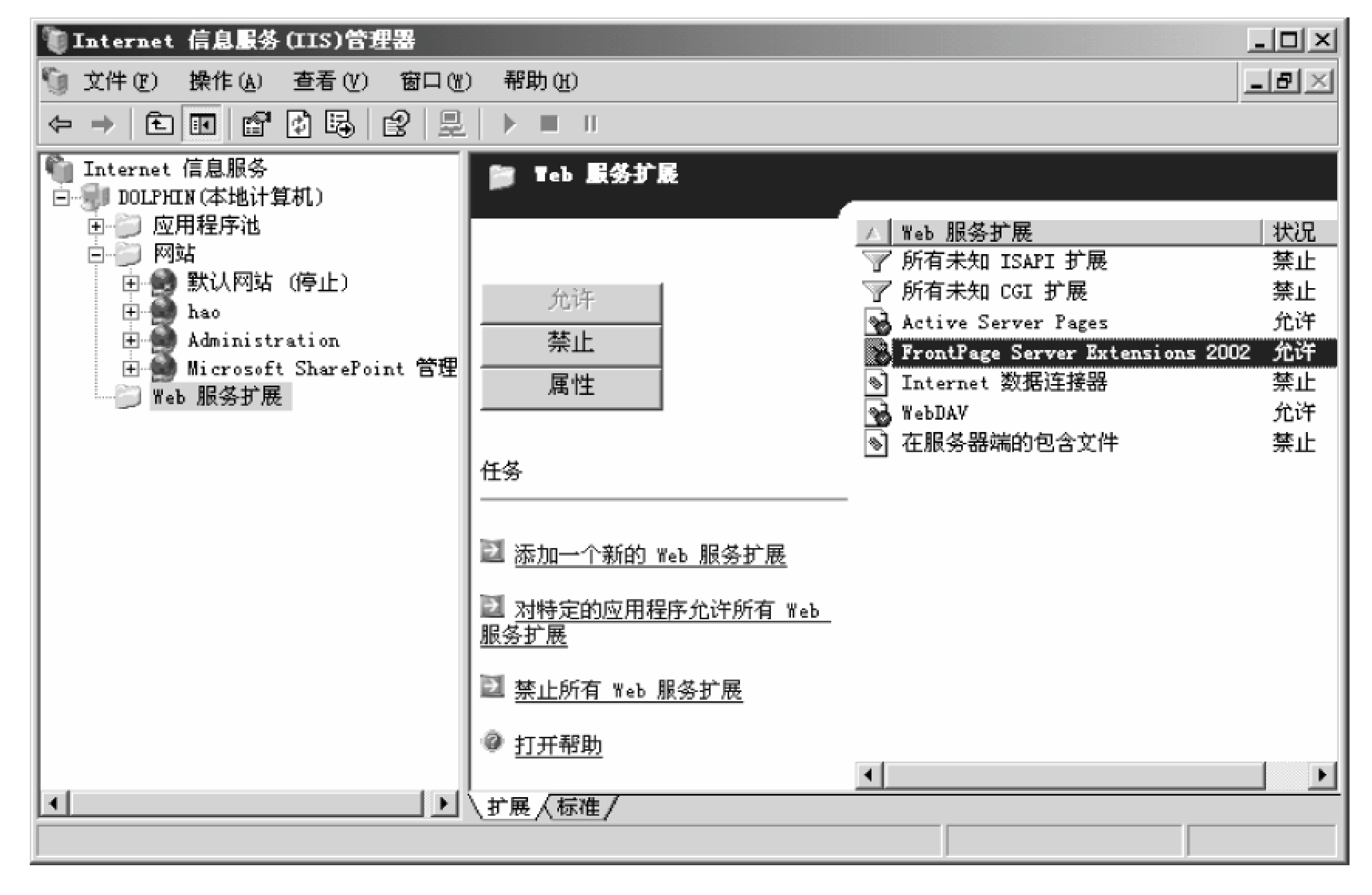



图 4-10　Web 服务扩展的配置

(4) 在 IIS 控制台中，右击要远程管理的站点，执行“属性”命令，查看站点属性，显示“Server 2002 Extensions”选项卡。然后，在站点上右击鼠标，指向“所有任务”，执行“配置 Server 2002 Extensions”命令。

(5) 在 IIS 控制台中，在“站点属性”对话框中，选择“目录安全性”选项卡，可以设置站点维护的安全账号等信息。

当对 Web 服务器和相应的站点进行上述的一系列设置后，用户就可以对服务器进行远程管理(Web 接口为 http://网址：6859/)或利用 FrontPage 对站点进行远程维护。在网站视图下，单击“远程网站”选项卡，然后单击窗口中的“远程网站属性…”，打开“远程网站属性”对话框，如图 4-11 所示。

根据安装的 Windows 服务组件不同，远程 Web 服务器分为四种类型，以支持 Web 站点的远程维护。下面以“FrontPage 扩展或 SharePoint Services”服务类型为例，介绍 Windows 平台中网站的远程管理。




图 4-11 "远程网站属性"对话框

在 Windows 平台上，如果远程 Web 服务器上安装了 FrontPage 扩展或 SharePoint Services，可以利用 FrontPage 对 Web 站点进行远程维护。在"远程网站属性"对话框中，选择"FrontPage 扩展或 SharePoint Services"单选钮，然后在远程网站位置下面的文本框中输入远程网站的网址和端口号(见图 4-11)。

系统提示输入远程 Web 服务器的账户和密码，输入完毕后，显示本地网站和远程网站目录结构，如图 4-12 所示。

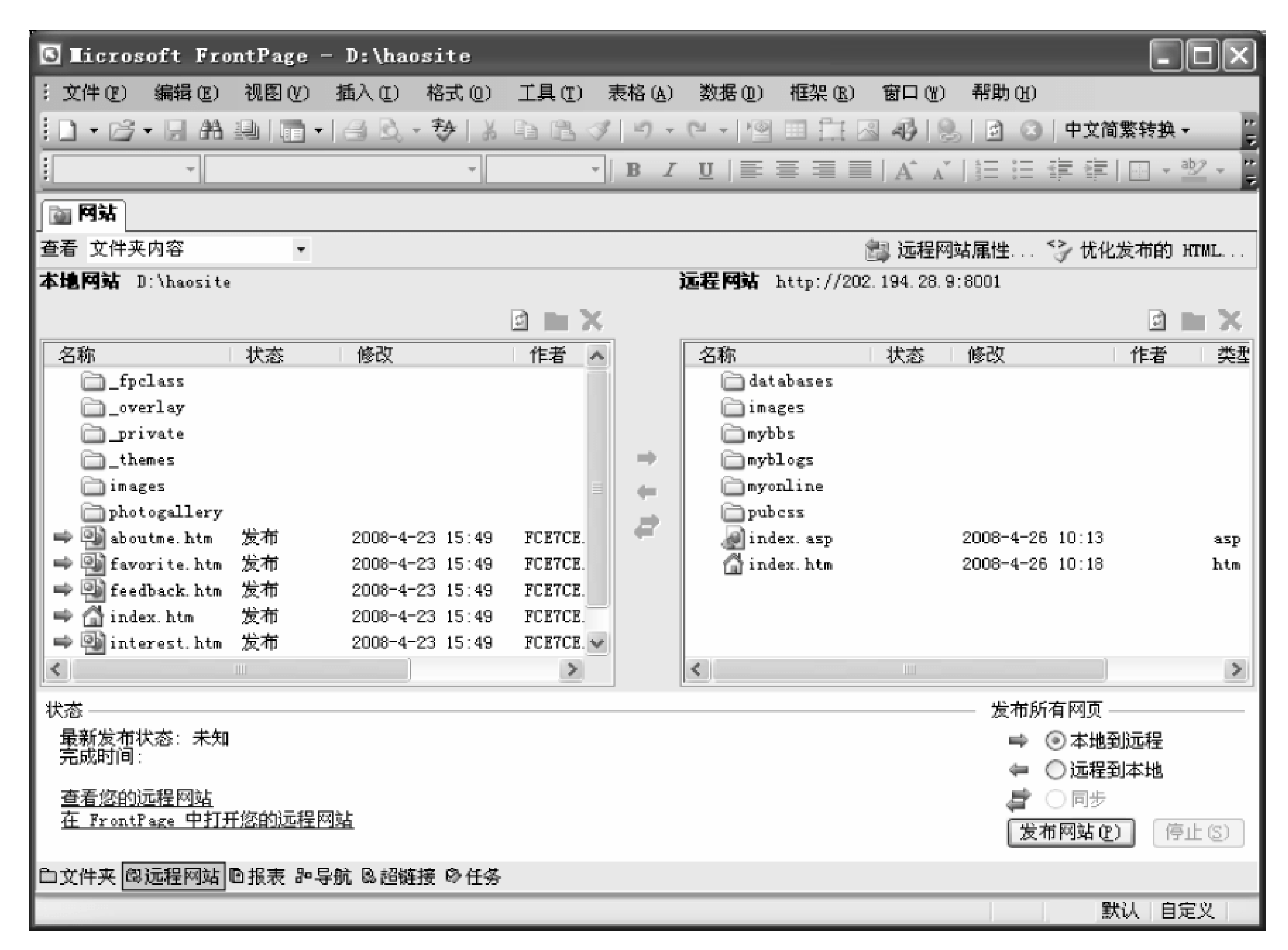



图 4-12 使用 FrontPage 远程维护网站

如果服务器不支持 FrontPage 扩展，则可选择 WebDAV（远程服务器支持“分布式创作和版本控制”）或 FTP 方式，进行站点的远程维护，操作方法类似，详细介绍略。

最后需要说明的是，除了使用 FrontPage 进行远程站点的维护外，用户还可以通过在服务器上安装终端服务，使用远程桌面链接到服务器，进行站点的维护；或者在 Web 服务器上安装 FTP 服务，设置 FTP 站点的根为 Web 站点的根，通过 FTP 来实现 Web 站点的维护。

### 4.2.3 新建网页

一个网站对应一个主目录，里面包含了大量的网页文件，这些网页通常按照网页功能组织在不同的文件夹中。因此，一个网站和传统的软件开发中的一个项目（Project）类似，每一个网页文件都是网站的一部分。

在 FrontPage 2003 中，要新建一个网页，通常经过以下几个步骤：

（1）在“文件”菜单中，执行“打开网站…”命令，选择要打开的网站主目录。因为每一个网页都是网站的一部分，因此，新建网页前，首先应打开网页隶属的网站。

（2）在“文件”菜单中，执行“新建…”命令，在客户区右侧显示“新建”任务窗格。

（3）在“新建”任务窗格中，单击“空白网页”超链接，将新建一个空白网页。单击“其他网页模板”，则打开“网页模板”对话框，显示 FrontPage 提供的默认模板。用户可以用这些模板来建立相应的网页，在右下角的预览区域可以看到选中模板的外观。

新建网页后，系统将生成一个 HTML 文档内容的基本框架，在“代码”模式或“拆分”模式下，可以看到相应的 HTML 代码内容，如图 4-13 所示。

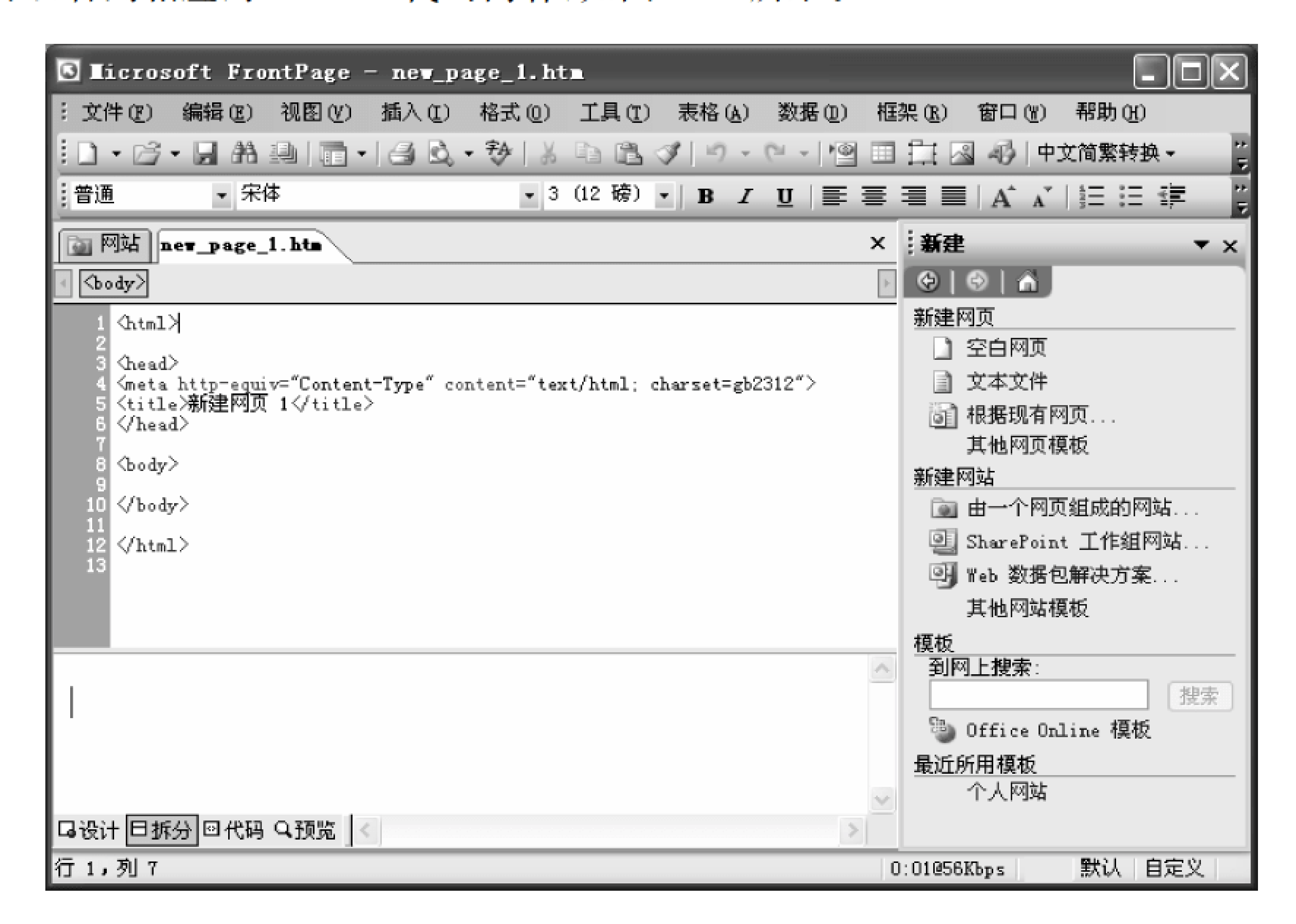



图 4-13　新建网页系统自动生成的 HTML 代码

从上述代码也可以看到,在文档的<body>…</body>标记之间还没有内容,具体内容根据用户的需要来设计。

使用 FrontPage 编辑网页,实际上就是利用 FrontPage 的菜单命令,在网页中完成输入文本、插入图片、建立超链接、插入表格、插入表单等操作,FrontPage 将自动生成对应的 HTML 代码,从而简化 HTML 文档的手工编写任务。

(4) 网页编辑完成后,在“文件”菜单中,执行“保存文件”命令,或者直接单击“常用”工具栏中的“保存文件”按钮,保存当前网页。

如果网页文件是第一次存盘,则打开“另存为”对话框,在“另存为”对话框中,选择网页的存储位置、文件名和保存类型。

## 4.3 网 页 编 辑

当网页文件创建后,接下来就是网页内容的编辑了。网页文件是一种符合 HTML 规范的纯文本文件,可以使用任意的文本编辑器进行编辑,但是这样的编辑需要非常熟悉 HTML 规范,并且效率很低,容易出错。因此,对网页的编辑通常使用 FrontPage、Dreamweaver 等可视化的网页制作工具来完成。

使用 FrontPage 的工具进行网页编辑,实际上就是利用 FrontPage 的菜单命令,在网页中实现输入文本、插入图片、建立超链接、插入表格、插入表单等内容,FrontPage 将自动生成对应的 HTML 代码,从而简化 HTML 文档的手工编写工作。

### 4.3.1 输入文本内容

文字是 Web 页的主要内容,在 HTML 规范中,有很多标记用于对文本进行标记,如<p>、<font>等,使用这些标记可以实现文本显示的格式化,即设定文本在浏览器中的显示效果。

在 FrontPage 中,文本的输入和格式化非常简单。在设计模式下,在编辑区域进行输入,再通过“格式”工具栏对文本进行格式化,或者通过“超链接”工具按钮将文本设置为超链接(<a>…</a>标记)。

例如,在设计模式下输入一行文本“泰山”,并进行简单的格式化(居中、加粗、字号、颜色),如图 4-14 所示。

在拆分模式下或代码模式下,可以看到在<body>…</body>标记内增加了如下代码:

```
<p align = "center"><b><font size = "6" color = "#FF0000">泰 山</font></b></p>
```

这段代码是 FrontPage 根据我们的输入自动生成的,可见通过 FrontPage 进行网页设计要比用“记事本”等程序手工书写 HTML 代码方便和快捷得多,并且也不容易出错,这正是 FrontPage、Dreamweaver 等工具的优势所在。

另外,使用 FrontPage、Dreamweaver 等工具,用户也可以从记忆大量的 HTML 标记的任务中解脱出来,从而把重点放在网页布局的设计和业务逻辑的设计中。

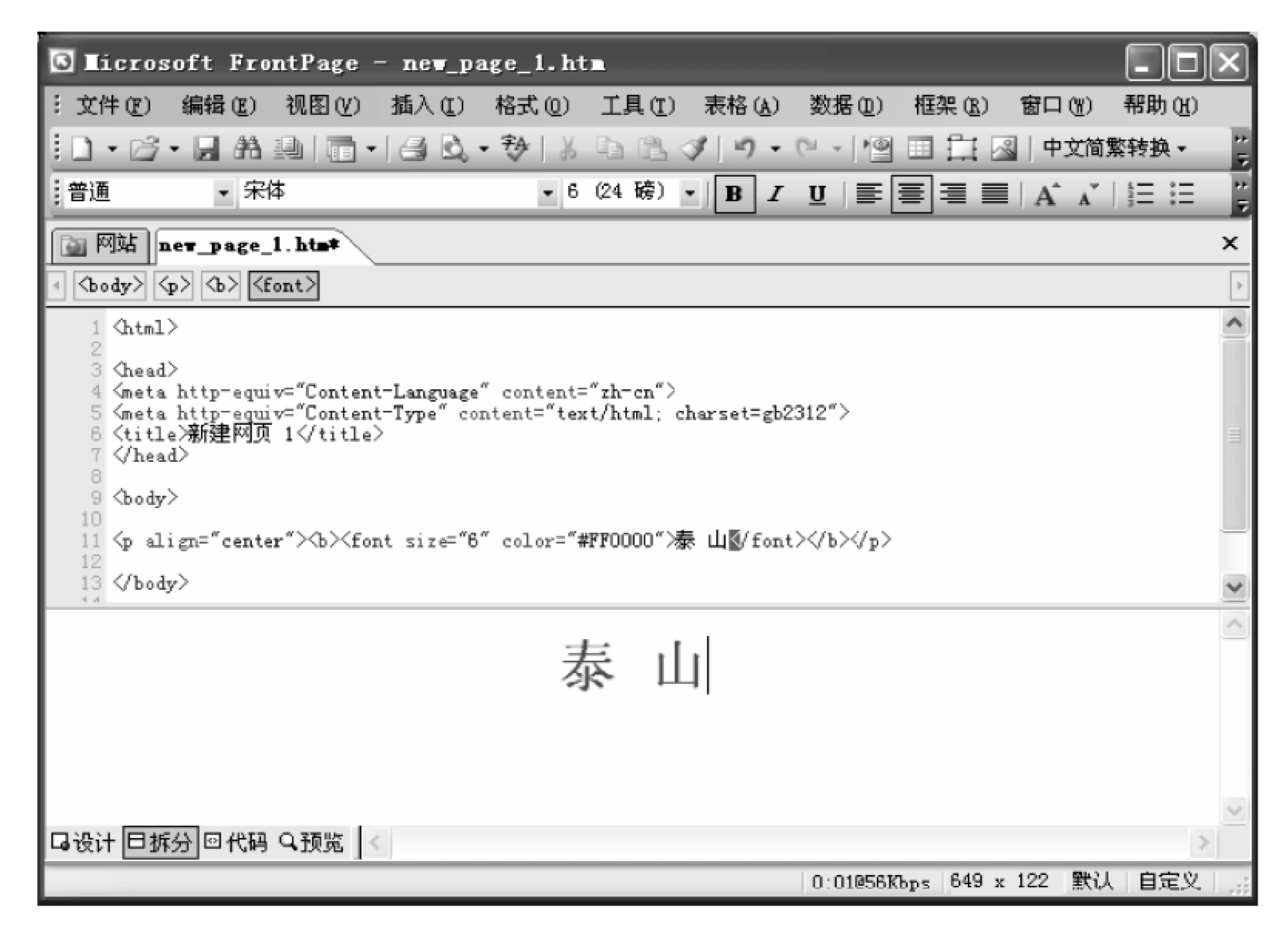



图 4-14　文本内容的输入与格式化

## 4.3.2　插入图片

在网页制作中经常需要插入一些图片，以达到丰富信息内容和美化页面的双重功能。在网页中插入图片，即通过<img>标记来指向一个图片文件，为了页面布局的需要，通常还需要设置<img>标记的属性。

要使用 FrontPage 在网页中插入一幅图片，具体操作步骤如下：

(1) 将插入点定位到要插入图片的地方(本例定位到文字的开始)。

(2) 选择菜单“插入”、“图片”、“来自文件”，打开“图片”对话框。选择要插入的图片文件，单击“插入”按钮，则图片自动以无环绕样式的方式插入到网页中。

(3) 设置图片属性。设置图片由于图片的高度和文字不一致，接下来应该设置图片的环绕方式。在 FrontPage 2003 中，右击图片，在快捷菜单中，执行“图片属性”命令，打开“图片属性”对话框，设置图片环绕样式。

设置完成后的页面如图 4-15 所示。

单击“拆分”或“代码”显示模式，可以看到在<body>…</body>标记内增加了一行类似下面的代码：

```
<img src = "file:///c:/xxx/tai.jpg" width = "281" height = "175" align = "right">
```

这是图片的绝对路径，在 Web 应用开发中，要避免使用绝对路径。因为当网页文件或者存储图片的目录发生改变，在打开网页时就会出现找不到图片文件的错误，图片在浏览器中不能被显示。

为什么会产生上述问题呢？这是因为如果是在一个新建的网页中插入图片，因为当前

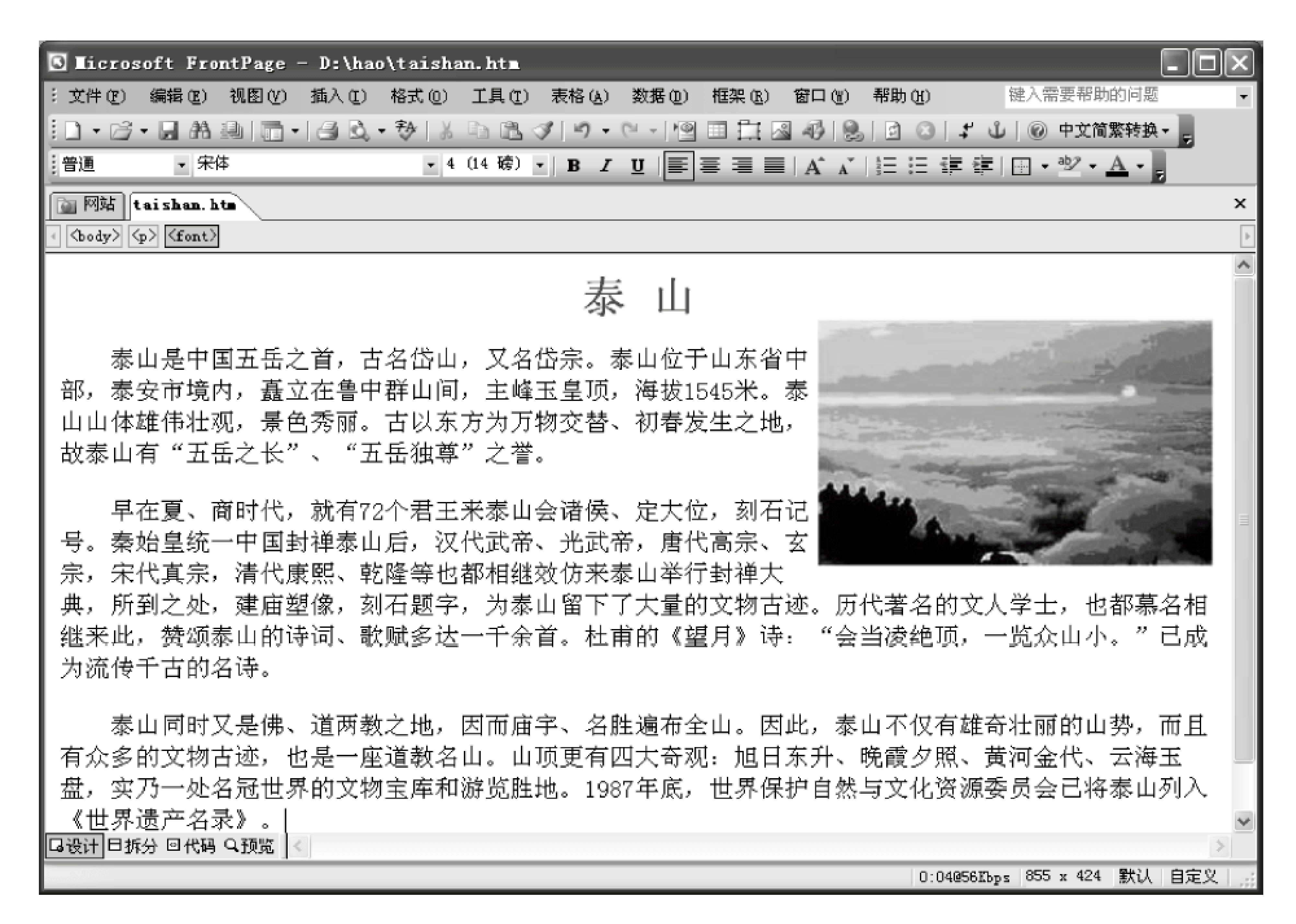



图 4-15 在网页中插入图片

网页尚未保存，因此无法判断网页文件和图片文件之间的相对路径，只能采用绝对路径的方式插入。此时，如果将新建的网页先保存一次，再插入图片时，在 HTML 代码中就会使用相对路径。或者，即使不是先插入了图片，只要保存一次 HTML 文件，原先的绝对路径也会变成相对路径。但是，如果 HTML 文件和图片不在一个驱动器上，将不能转换成相对路径。

最后要记住，既然是一个 Web 应用，所有的网页和用到的图片以及其他各种文件都应该保存在 Web 站点的主目录下，或者主目录下面的子目录下。所有这些文件共同构成了一个 Web 应用，所有的引用都应该是相对路径，这样当 Web 应用，即该站点被复制到其他的目标位置时，才不会出现找不到网页或图片的错误。

图片插入以后它在页面上的大小、位置等可能并不理想。要使得图片符合用户的要求，需通过设置图片属性来完成。图片属性的设置分别对应了<img>标记的不同属性，用户可以通过“拆分”或“代码”视图查看生成的 HTML 代码。此外，将图片直接插入网页中，要进行精细的布局是很困难的，要对网页进行布局，通常需要使用表格来完成。

### 4.3.3 建立超链接或书签

使用 FrontPage 在网页中建立超链接非常简单，只需要选定超链接的文本或图片，然后，单击工具栏上的“超链接”按钮，即可打开一个“插入超链接”对话框，在超链接对话框中输入被链接的 URL 即可完成。

（1）定义书签

书签是文档中的一个特定标记，便于打开网页定位到特定的位置。使用 FrontPage 在网页中定义书签，可按照下面步骤操作：

在网页中选定文字，然后在“插入”菜单中，执行“书签…”命令，打开“书签”对话框；在“书签”对话框中，显示书签名称和网页中已经定义的书签列表，输入书签名，然后单击“确定”按钮，即可定义一个书签，生成的 HTML 代码形式如下：

<a name = "书签名">书签文本</a>

在定义书签时，可以不选中任何文本，这样只是在插入点处定义一个书签。

（2）文本超链接

使用 FrontPage 在网页中建立文本超链接，操作步骤如下：

① 选定要建立超链接的文本。

② 选择菜单“插入”、“超链接”，或直接单击常用工具栏上的“超链接”按钮，打开“插入超链接…”对话框，如图 4-16 所示。

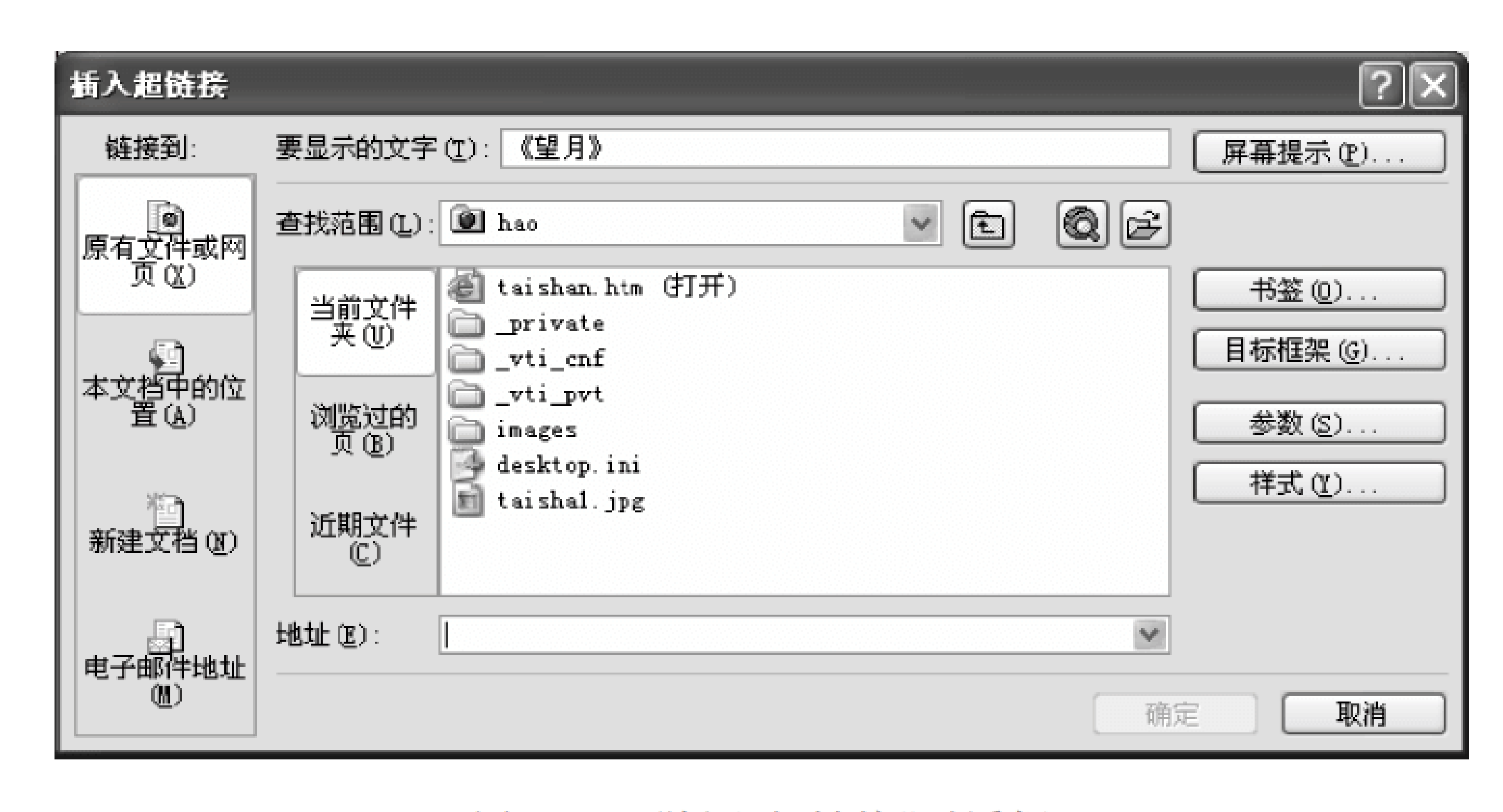



图 4-16　“插入超链接”对话框

③ 在“插入超链接”对话框中，在“链接到”下面的列表中，有四种主要的链接到目标类型，包括：

- 原有文件或网页：通过 Web 浏览器查找和选择被链接的网页。
- 本文档中的位置：即链接到网页中的一个书签，通过“选择文件”对话框在本地机上查找和选择被链接的文档。同样需要注意的是文件和被链接的 HTML 文档之间的路径问题。

如果是同一个 Web 应用中的 Web 页面，必须使用相对路径，方法类似于插入图片。如果是当前 Web 应用以外的 Web 页面，需要给出完整的 URL 地址。

- 新建文档：新建一个空白文档并且与之建立超链接。
- 电子邮件地址：与一个 E-mail 地址建立超链接。

④ 设置目标框架和书签，单击右侧的“书签…”按钮，可以显示选择的网页中定义的书签。单击右侧的“目标框架…”按钮，可以指定单击超链接时打开网页的窗口，分别对应<a>标记中的 target 属性和 name 属性。

⑤ 通过“样式”按钮，可以设置<a>标记的 class 和 ID 属性。

当上述设置完成后，单击“确定”按钮，完成超链接的建立，即建立一个文本超链接。自动生成对应的 HTML 代码，形式如下：

```
<a href = "poems/wangyue.htm" target = "_blank" >《望月》</a>
```

当文本对应一个超链接时，文本将以特殊的颜色显示，超链接文本默认显示为蓝色带下划线的文字。显示网页时，当鼠标指针指向该文本时，鼠标指针将变为手形指针。

(3) 图片超链接

使用 FrontPage 也可以为图片建立超链接，具体操作步骤如下：

首先，选中将要建立超链接的图像；然后选择菜单“插入”、“超链接”，打开“超链接”对话框；接下来的设置与文本超链接的操作相同。

### 4.3.4 图像地图

在 HTML 规范中，图像地图是一种常用的页面导航界面，对于热点的边界点(x，y)坐标，手工定义几乎不可能，因此，在页面中使用图像地图都是通过 FrontPage 等页面制作工具完成的。使用 FrontPage 建立图像地图，具体步骤如下：

① 在“插入”菜单中，指向“图片”，执行“来自文件…”命令，插入一幅图片。

② 单击图片，则在 FrontPage 窗口的底部显示图片工具栏。如果不显示图片工具栏，可在图片上右击鼠标，在快捷菜单中执行“显示图片工具栏”命令，即可显示图片工具栏，如图 4-17 所示。

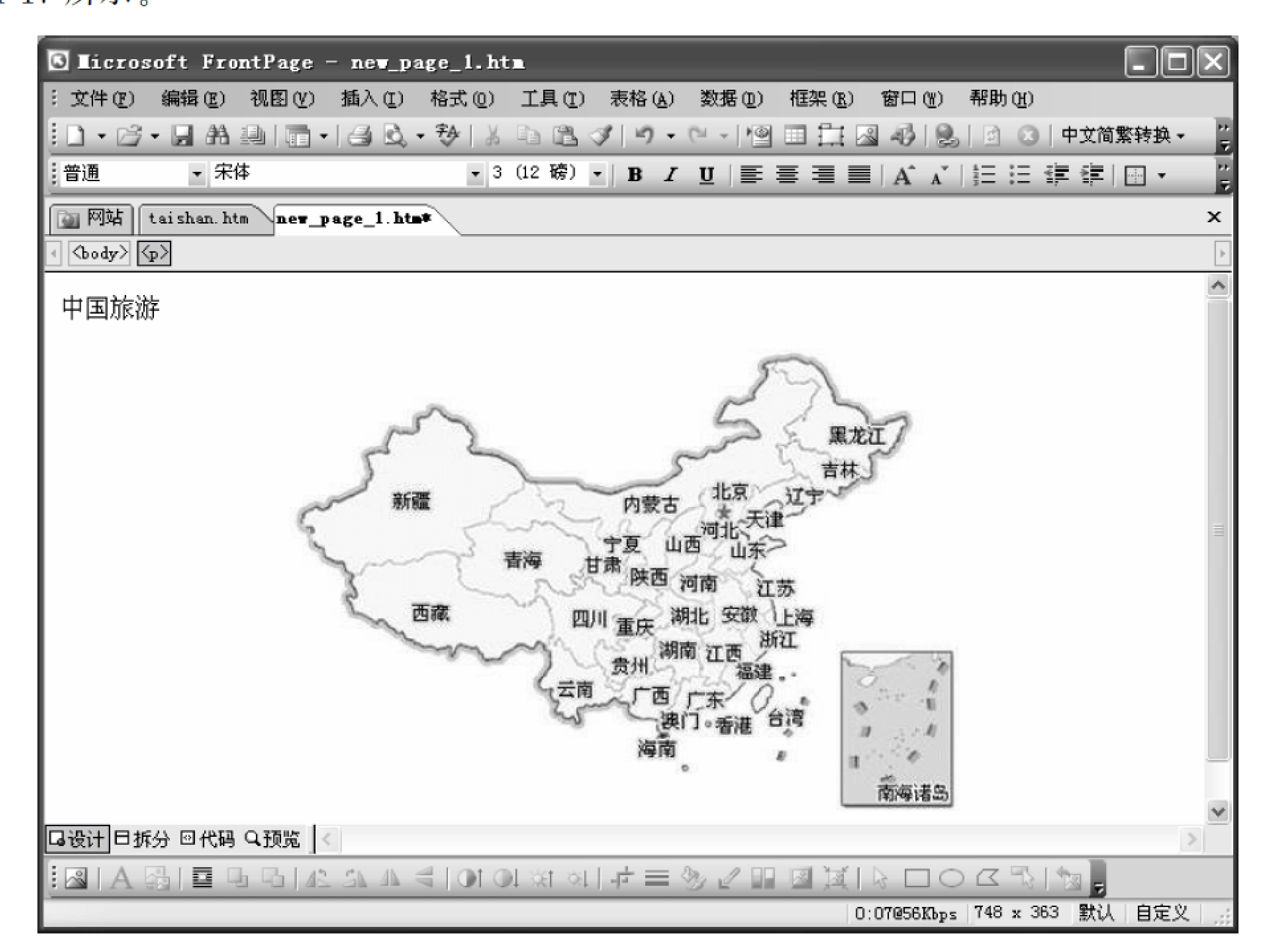



图 4-17 定义图像地图

定义图像的热点可通过“图片”工具栏完成，其中有四个定义热点的工具：长方形热点、圆形热点、多边形热点和突出显示热点。

定义热点的方法很简单：选中图像，这时上述的四个工具变为可用；然后按住鼠标左键在图像上的目标区域拖动即可。对于多边形热点，则可以逐个顶点位置点击，直到形成一个封闭的区域为止。热点区域建立后，将自动打开“超链接”对话框，输入要链接的 URL 即可。

网页显示时，图像上的热点区域是不可见的，但是在热点设置时按下“图片”工具栏上的“突出显示热点”按钮，这样在显示网页时，如果单击热点区域，可以看到区域的外框。

自动生成如下所示的 HTML 代码：

```
<map name = "FPMap0">
<area href = "city/beijing.htm" shape = "circle" coords = "286, 100, 17">
<area href = "city/jinan.htm" shape = "polygon" coords = "279, 129, 295, 111, 323, 122, 305,
136, 287, 145">
</map>
<img border = "0" src = "chinamap.gif" width = "433" height = "270" usemap = "#FPMap0">
```

## 4.3.5　插入表格

在网页设计中，表格不仅仅用来显示数据，而且表格还是进行网页布局的重要手段。使用表格，可对各种网页元素的位置进行精确控制，例如文字和图片混排，从而使设计的网页布局整齐、美观。

使用 FrontPage，在网页中插入表格，可以按照下面步骤操作：

① 在网页中插入表格，选择菜单“表格”，指向“插入”，打开“插入表格”对话框，可输入要插入表格的行数和列数，设置布局、边框、背景等属性，即可插入一个表格。

② 表格编辑，插入到网页上的表格，可以进行一系列的编辑操作，如调整行高和列宽，单元格的合并与拆分，删除行、列或单元格，插入行、列或单元格等。

③ 设置表格属性，将插入点放在表格当中，右击鼠标，从快捷菜单中选择“表格属性”，打开“表格属性”对话框，如图 4-18 所示。

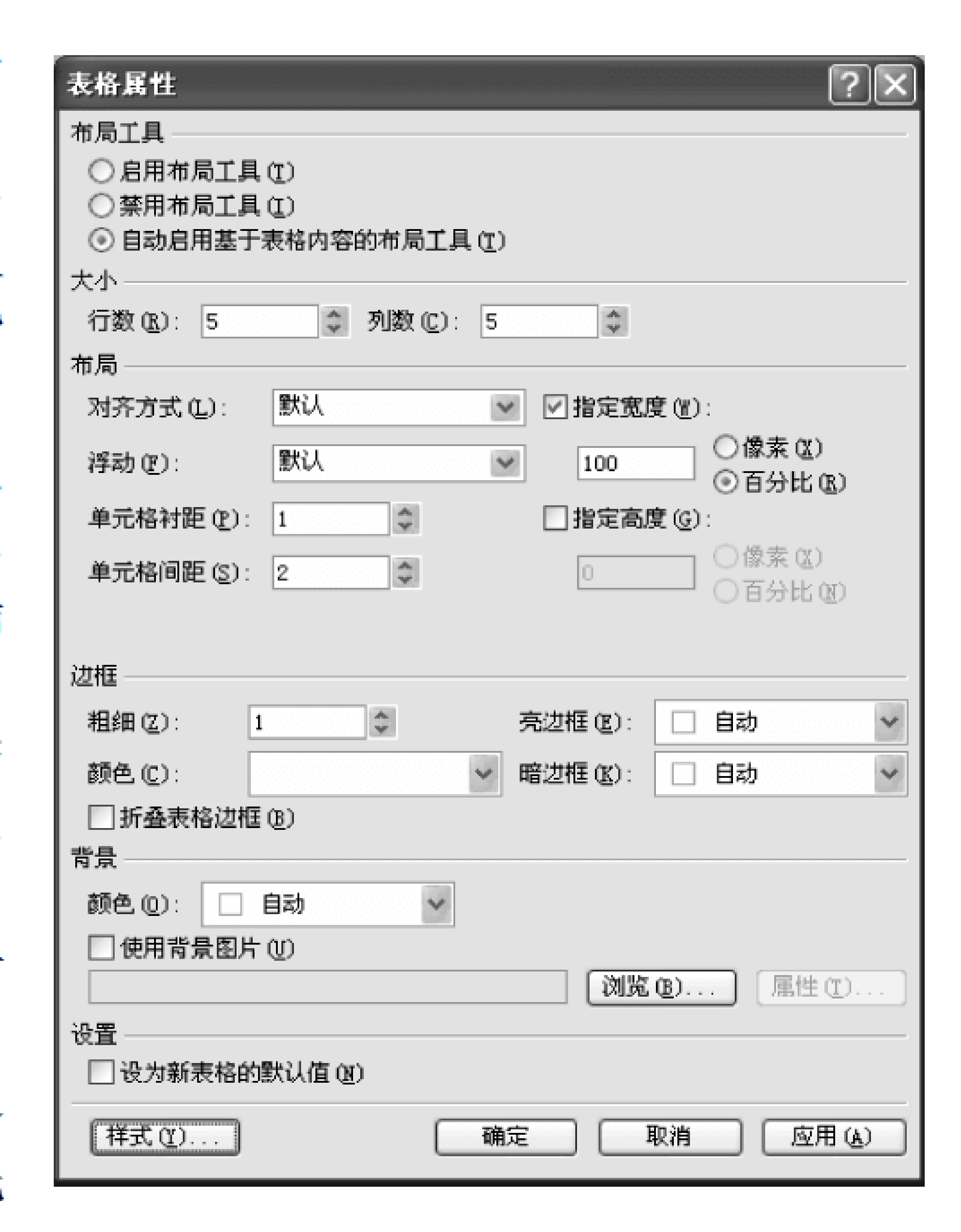



图 4-18　FrontPage 2003“表格属性”对话框

在“表格属性”对话框可进行以下几个方面的设置：

- 布局属性：包括表格的对齐方式、表格宽度、单元格间距（cellspacing）、单元格衬距（cellpadding）。其中，单元格间

距指单元格之间的距离，默认值为 2，可设值为 0；单元格衬距指单元格中文字到表格线的距离，默认值为 1。

- 表格边框设置：设置表格边框的粗细和颜色。
- 表格背景：可以设置表格所用的背景色或背景图片。

在网页中，如果要使表格随着浏览器的窗口变化而变化，在表格属性中，选择“指定宽度”复选框，然后选择“百分比”单选钮。此时，如果表格的宽度大于浏览器窗口的宽度时，表格内容将换行，可能影响表格的外观。如果在表格属性中，选择“指定宽度”复选框，并且选择“像素”单选钮，则表格的显示将不随浏览器窗口的变化而变化。

④ 设置单元格属性，当需要对单元格进行特殊设置时，可以通过“单元格属性”对话框完成。

将插入点放到单元格中，或者选定要设置的单元格，然后右击鼠标，从快捷菜单中选择“单元格属性…”命令，打开“单元格属性”对话框，可以对一个单元格进行设置，这些设置将影响<td>标记的属性。

通过 FrontPage 的“拆分”视图或“代码”视图，可以看到生成的 HTML 代码，表格属性和单元格属性分别对应了<table>标记、<tr>标记以及<td>标记的有关属性，如图 4-19 所示。

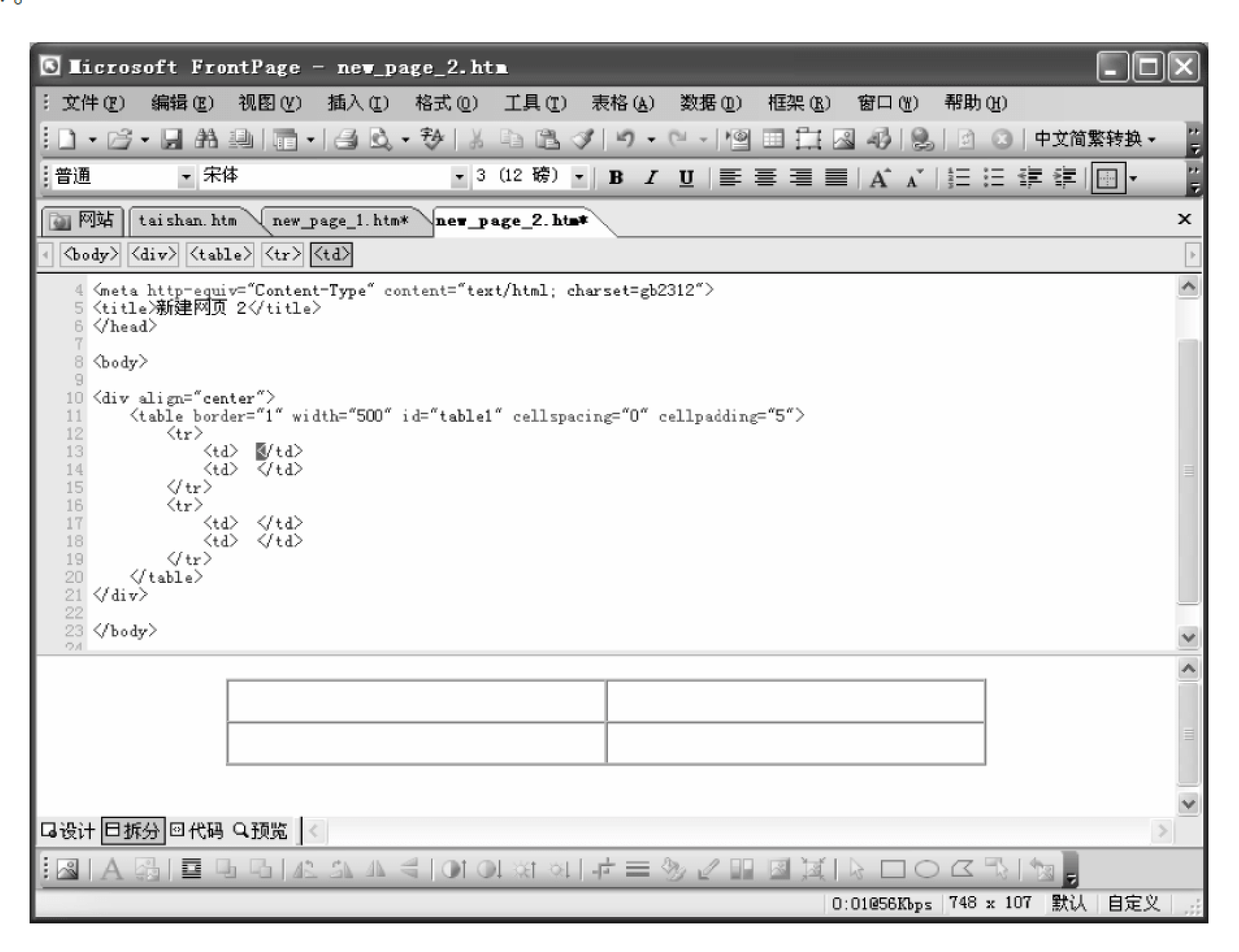



图 4-19 插入表格及其属性设置

上述表格属性布局设置为：

对齐方式(居中)，单元格衬距(5)，单元格间距(0)，指定宽度，像素(500)。

## 4.3.6　插入表单

对于一个负责用户交互的网页，其主要内容就是表单。使用 FrontPage 在网页中插入表单，具体操作步骤如下：

① 在“插入”菜单中，指向“表单”，选择要插入的表单或者其中的控件。

② 调整表单布局。表单一般和表格联合使用，通过表格设置表单布局。FrontPage 有时候使得这些标记交叉，为此，可以在“拆分”或“代码”视图中，手工调整代码。例如，将<form></form>标记对调整到<table></table>标记对的外面。

在使用 FrontPage 插入表单或控件时，如果插入了多个<form>标记，也可以手工地删除，只保留一个<form></form>标记对，手工把所有的控件标记都移动到<form></form>内部。

③ 设置表单属性，在表单上右击鼠标，在快捷菜单中执行“表单属性…”命令，打开“表单属性”对话框，如图 4-20 所示。

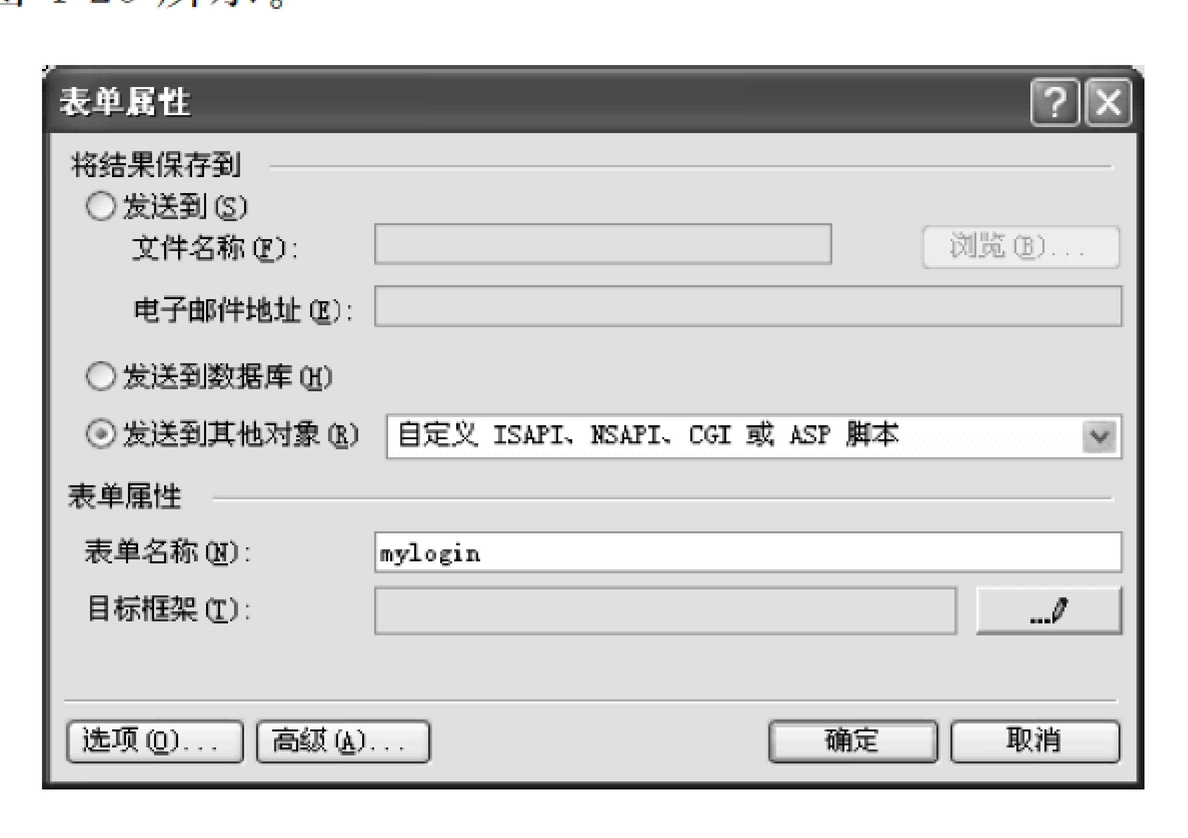



图 4-20　设置表单属性

在“表单属性”对话框中，可以进行各种设置，即对应<form>标记的属性设置。

④ 设置表单域属性，在表单域上右击，在快捷菜单中执行“表单域属性…”命令，打开“表单域属性”对话框。不同类型的表单域，其属性对话框也不相同，文本控件对应的表单域属性对话框如图 4-21 所示。

图 4-21　设置表单域属性

在“表单域属性”对话框中,输入相应的属性值,为脚本编程需要,一般要设置表单域名称等属性,最后单击“确定”按钮。

上述设置结束后,生成的网页及其 HTML 代码示例如图 4-22 所示。

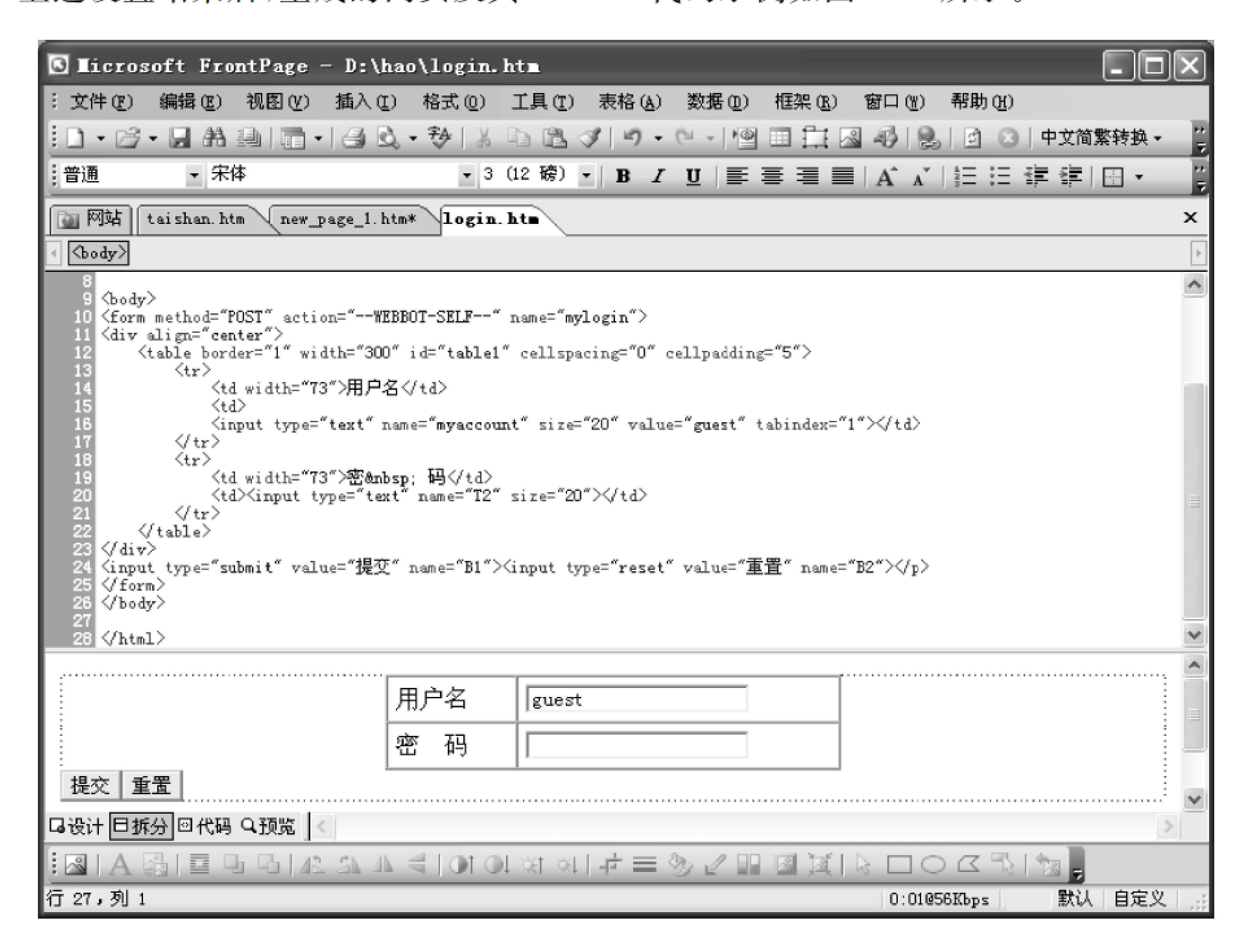



图 4-22 插入表单及其属性设置

## 4.4 设置标记属性

在 HTML 规范中,每个标记都含有不同的属性,通过设置标记属性可以修改标记的默认显示样式。标记的属性众多,很难记忆。利用 FrontPage,在“设计”视图下,通过一些标记的属性对话框可以进行标记的属性设置。除此之外,还可以在“代码”视图下,通过 FrontPage 的智能感知技术设置标记属性。

### 4.4.1 使用 IntelliSense 技术

现在大多数的开发环境都使用智能感知(IntelliSense)技术为用户提供帮助。所谓 IntelliSense,是指当用户编辑到一个对象时,系统能动态地显示当前对象的方法、属性名列表,从而保证用户输入的正确性,或从中选择输入。在 Boland 的开发环境中,类似的技术称为 CodeInsight。下面举例说明:

在一个网页中,假定要设置一个超链接的属性。单击“代码”视图,将插入点定位到<a>

标记，按空格键，则自动打开一个窗口，显示<a>标记的所有属性，包括一般属性和事件属性，如图 4-23 所示。

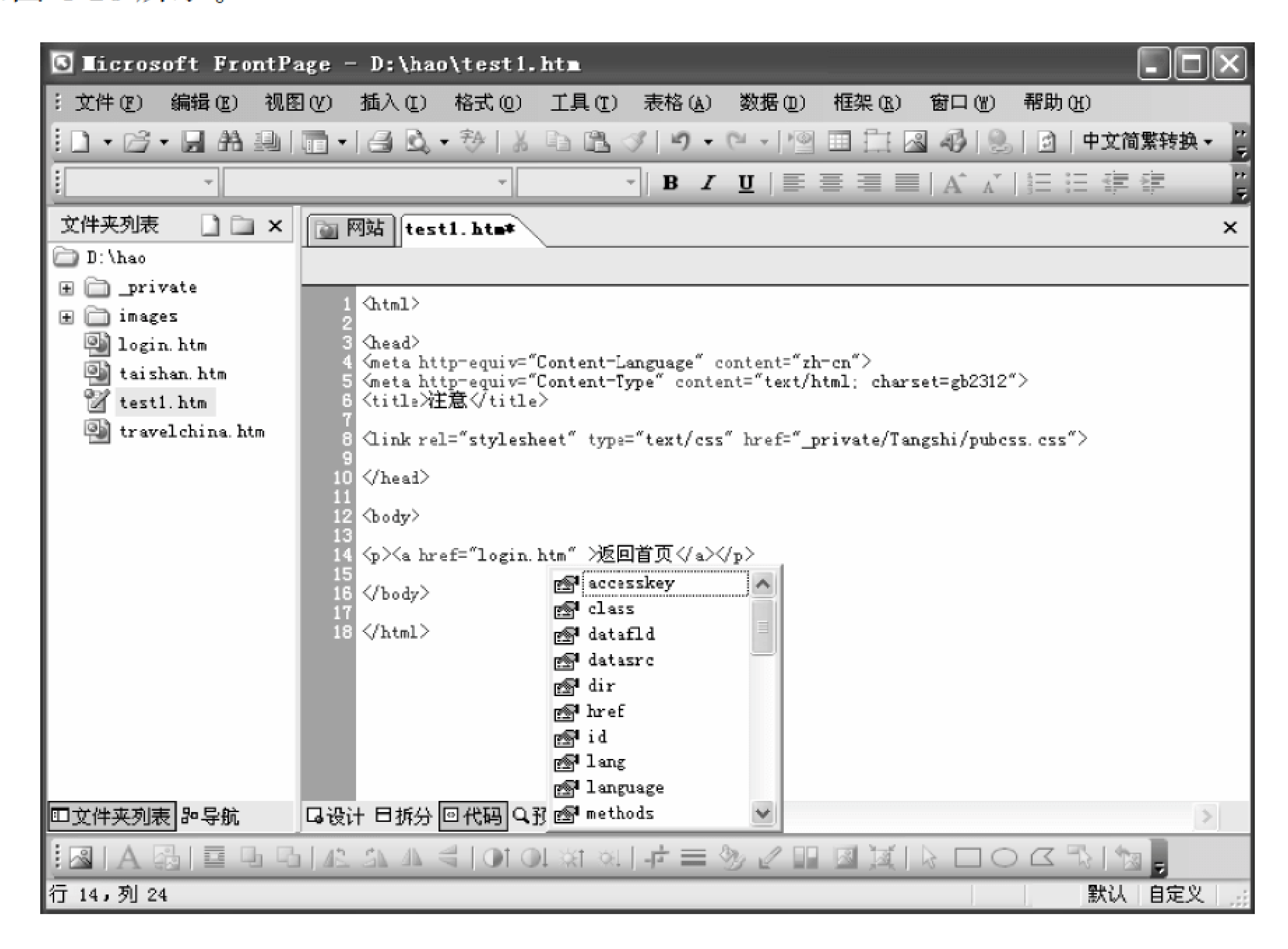



图 4-23　使用 IntelliSense 技术

现在大多数软件都使用了 IntelliSense 技术，有的软件默认情况下不打开智能感知，需要对软件进行设置，开启 IntelliSense 功能。

## 4.4.2　使用行为面板

标记中的事件属性对应的事件，又称为行为，即用户对所标记对象的操作行为。每个行为事件往往都对应一个函数，即标记的事件属性取值通常是一个函数。在 FrontPage 中，有许多内置的行为函数，使用这些函数可以简化用户的编码。具体操作如下：

（1）要设置标记的行为，首先选择“设计”视图，在“格式”中执行“行为…”命令，打开“行为”任务面板。

（2）单击“插入”，在下拉列表中可以选择一个行为。则在下面的事件列表中显示行为对应的默认事件。

（3）如果需要修改行为对应的事件，可以单击事件，则显示当前对象的可选事件列表，选择一个事件作为当前行为的激活事件即可。

执行以上操作步骤，结果如图 4-24 所示。

转到“代码”视图，可以看到标记中增加了相应的事件属性，并可以看到自动生成的脚本程序代码。

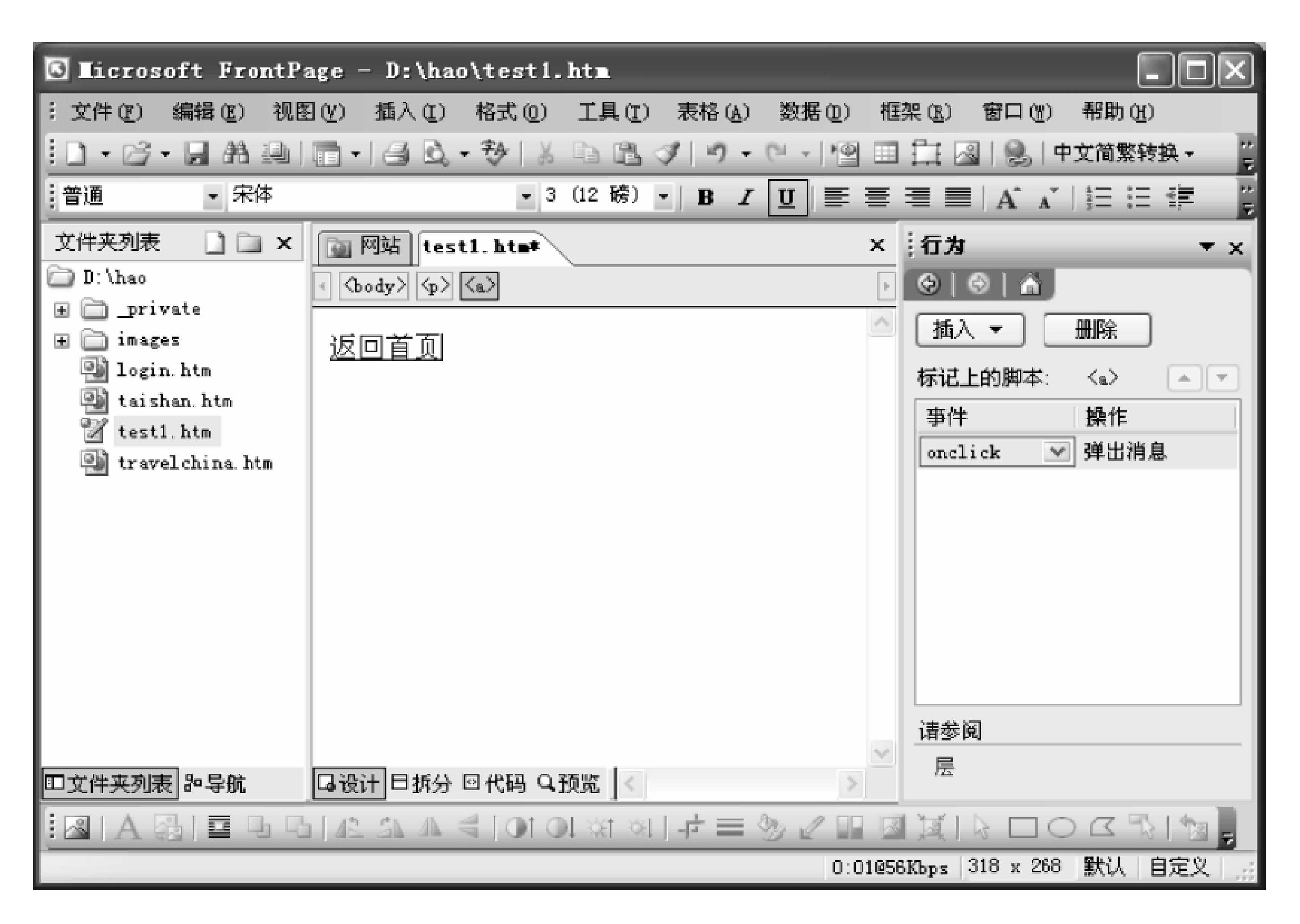


图 4-24　设置标记的事件属性

## 4.5　定义和使用样式

HTML 允许用户在标记中使用 class 属性修改标记的默认显示样式，手工定义样式需要记住很多的属性及取值，非常麻烦。在 FrontPage 中，提供了修改样式和新建样式类功能。为了在多个页面中实现样式的共享，可以将用户自定义样式存储为样式表 CSS 文件。

### 4.5.1　定义样式

如果要在一个网页中使用样式，在 FrontPage 中，可以通过执行“格式”菜单中的“样式…”菜单命令来实现，具体操作步骤如下：

执行“格式”菜单中的“样式…”菜单命令，打开“样式”对话框，如图 4-25 所示。

在左下侧列表中可以选择“用户定义的样式”或者“HTML 标记”。

(1) 修改 HTML 标记默认样式。选择“HTML 标记”，则意味着修改默认样式，列表中将列出 HTML 规范的所有标记，然后选择一个样式，再单击“修改”按钮，修改所选的样式。

(2) 新建用户样式类。在样式列表中，选择“用户自定义样式”，单击“新建…”按钮，可以创建一个新的样式类，并对样式进行字体等属性的设置。

此时在 HTML 视图中，将看到文档的<head>…</head>内部生成类似如下的代码：




图 4-25　自定义样式对话框

```
<style>
<!--
a:active     { font-size: 10pt; color: #0000FF }
a:hover      { font-size: 10px; color: #FF0000 }
.warning     { font-size: 8pt; color: #FF0000; font-weight: bold }
-->
</style>
```

用户自定义样式后，可以在 FrontPage“格式”工具栏左边的样式列表中显示用户自定义的样式，用户可以用自定义样式格式化网页中的文档内容。此时，在生成的 HTML 代码中看到相应标记内增加了 class 属性。例如：<p class="warning">注意事项：</p>。

## 4.5.2　使用样式表文件

在网页内定义的样式，只是用于网页内部。为了实现多个网页之间一致的外观和自定义样式的共享，通常需要将样式定义保存为.css 文件，这样多个网页就可以共用样式，保证网站中网页风格的一致性。

建立样式表文件(.css 文件)有两种方法：

(1) 在“新建”任务窗格中，单击“其他网页模板”超链接，打开“网页模板”对话框，选择“样式表”选项卡，然后选择一个样式表模板，从而建立一个样式表文件。

(2) 新建一个普通的页面文件，自定义样式完成后，删除掉除<style>…</style>标记对以外的所有 HTML 代码，然后执行“另存为”命令，选择保存类型为“CSS 文件(.css)”。

当样式表文件定义完成后，在其他网页中就可以使用其中的样式了。在“格式”菜单中，执行“样式表链接…”命令，打开“连接样式表”对话框，选择一个.css 文件，例如 mystyles1.css，则在该网页的<head>…</head>内增加下述语句：

```
<link rel = "stylesheet" type = "text/css" href = "mystyles1.css">
```

# 4.6 Frame 框架和 IFrame 框架

在网页布局设计中,框架网页是经常使用的一种网页类型,它可以把浏览窗口分成几个区域,每个区域可以显示一个不同的页面,即对应一个单独的网页文件。

## 4.6.1 Frame 框架网页

### 1. 新建框架网页

在 FrontPage 2003 中创建框架网页可以通过专门的框架网页模板完成,方法如下:

(1) 在“文件”菜单中,执行“新建”命令,打开“新建”任务窗格,在新建网页区域单击“其他网页模板”超链接,打开“网页模板”对话框,如图 4-26 所示。




图 4-26 “网页模板”对话框

(2) 选择“框架网页”选项卡,则在左侧列表框中显示出了多个框架网页模板的图标,单击其中之一,可以在右侧的预览区显示该模板的布局样式。

(3) 双击选中的模板,或者选中模板后单击“确定”按钮,即可建立起相应的框架网页,图 4-27 就是选中“嵌套式层次结构”后所建框架网页的初始状态。

在这个框架网页中,浏览窗口被分成了 3 个框架,各个窗格显示不同的页面。可以看到,初始状态下每个框架中有两个按钮:

- 设置初始网页。所谓初始网页,是指浏览器第一次载入框架网页时该框架所示的网页。该按钮就是设置一个已经存在的网页作为该框架的初始网页。单击时可以打开“创建超级链接”对话框,选择一个网页文件或输入一个网页的 URL,然后单击“确定”按钮,即可在当前框架中打开所超链接的网页文档。
- 新建网页。单击该按钮时,可以新建一个普通的网页(空白页)作为初始网页。




图 4-27　新建框架网页示例

当新建或打开一个框架网页后，FrontPage 窗口下方的显示模式选项标签将发生以下变化：

（1）增加了“无框架”标签。当浏览器不支持框架网页时所要显示的内容。默认情况下将显示文本“此网页使用了框架，但您的浏览器不支持框架”。

（2）当前网页为框架网页时，“框架”菜单中的菜单项将变为可用，主要包括“拆分框架”、“删除框件”、“框架属性”等命令。

### 2. 拆分与删除框架

为了适应不同的需要，用户可以将一个网页框架随意拆分或合并，形成有自己风格的框架网页。方法如下：

（1）选中要进行拆分的框架，如在上述的网页示例中选中下方的框架。选中的框架将以蓝色粗线条包围。

（2）选择菜单“框架”、“拆分框架”，打开“拆分框架”对话框。

（3）选择“拆分成行”，或者“拆分成列”，即把当前框架窗格拆分成上下两个框架，或者拆分成左右两个框架。

如果要删除框架，选中要删除的框架，选择菜单“框架”、“删除框架”即可。

### 3. 改变框架窗格的大小

框架的大小是可以根据需要改变的。方法是：将鼠标指针指向框架，当指针变成↔形或者↕形时，按住左键拖动，达到满意的位置后放开。

#### 4. 设置框架属性

可以用以下方法设置框架属性：鼠标指针指向框架，右击，从快捷菜单中选择“框架属性”，打开“框架属性”对话框，如图 4-28 所示。

图 4-28 “框架属性”对话框

对话框中包含的内容有：

(1) 名称：框架的名称，对应<frame>标记中 name 属性，主要用于在超链接中指定<a>标记的 target。

(2) 初始网页：第一次显示框架网页时显示的网页。

(3) 框架大小：可以设置列宽和行高。有三种取值：相对(相对于同行或同列的其他框架)、像素和百分比(相对于窗口的高或宽所占的百分比)。如果一个框架是通栏的，则列宽为页面窗口宽度，不可作其他设置。

如果选择像素，则在浏览器窗口大小变化的时候，该框架将不会随着变化。

(4) 边距：框架中网页页面的边距，以像素为单位。

(5) “框架网页”按钮：单击此按钮时，可以打开“网页属性”对话框的“框架”选项卡，可以设置框架的宽度，以像素为单位。如果只设置框架的宽度，可以直接打开“网页属性”对话框完成设置。

(6) 在浏览器中是否可以调整大小：选中该复选框时，浏览框架网页时可以改变框架大小(用鼠标拖动框架)。

(7) 是否显示滚动条：有“需要时显示”、“从不”和“始终”三种选择。

(8) 可以修改框架的样式和格式。

#### 5. 设置超链接的目标框架

假设设计一个“唐诗欣赏”的框架网页，包含三个框架：目录区、网页标题区和诗文区。

我们希望单击目录框架的唐诗标题时，在诗文区可显示出该诗的正文来。下面是具体的实现步骤：

(1) 选中目录区的一首诗，例如“出塞”。

(2) 选择菜单“插入”、“超链接”，打开“创建超链接”对话框。

(3) 在对话框的下边有一个“目标框架”文本框，单击其后面的按钮，打开“目标框架”对话框，如图 4-29 所示。

(4) 在“当前框架网页”图示区域，单击链接网页显示的区域(本例中为右下角区域)，在“目标设




图 4-29 “目标框架”对话框

置”中自动填入该框架的名称，或从右侧“公用的目标区”列表中选择。

（5）完成后单击“确定”按钮，回到“创建超链接”对话框中。可以看到，“目标框架”中填入了所选框架的名称。生成的 HTML 代码中在＜frame＞标记中添加了 target 属性。

（6）选择被链接的 HTML 文档，最后单击“确定”按钮。

按以上方法建立所有诗标题的超链接。完成后的网页如图 4-30 所示。

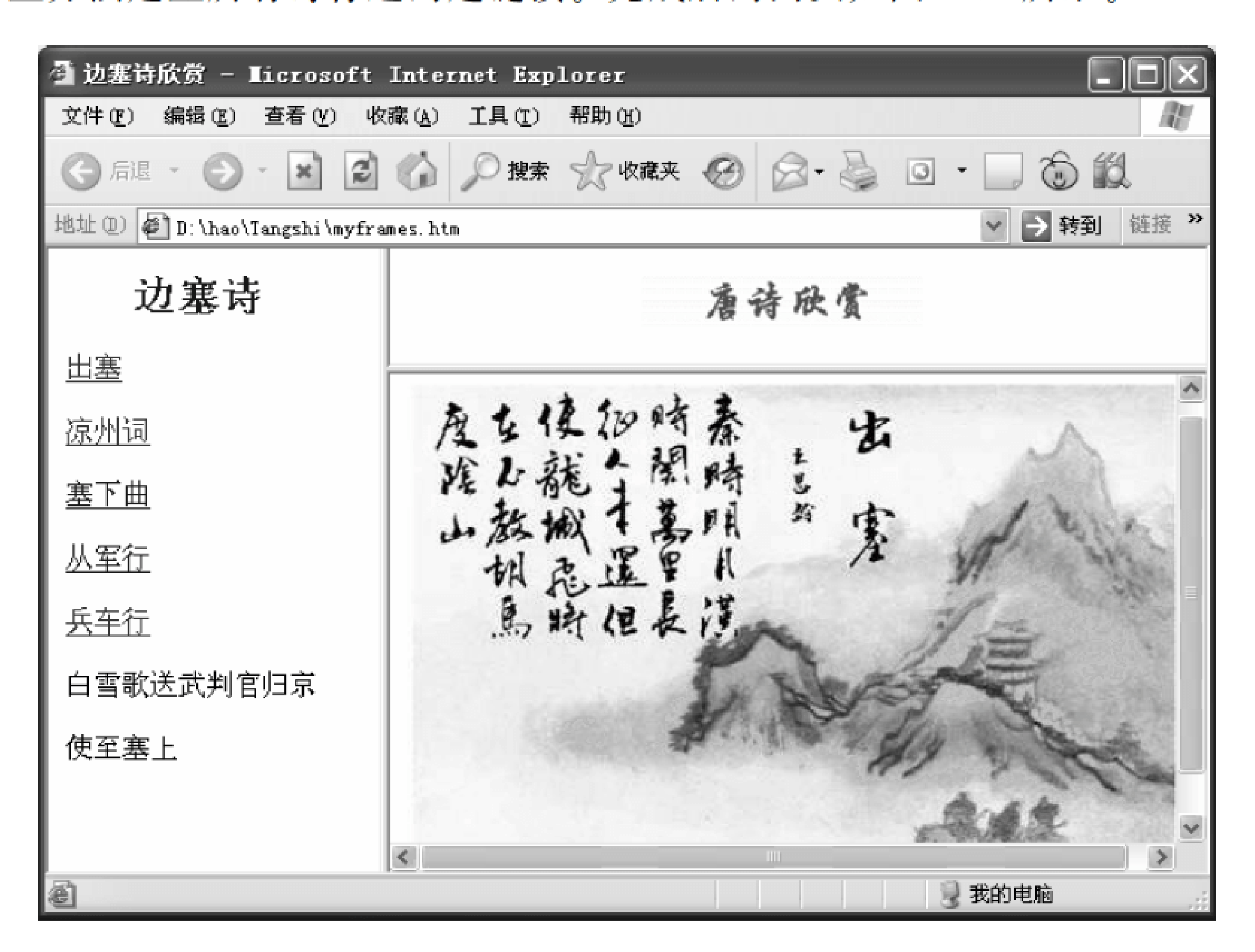



图 4-30　框架网页应用示例

通过框架网页以及后面介绍的客户端脚本程序、表单对象，可以较好地完成 Web 应用中页面布局和用户接口设计。

## 4.6.2　使用浮动框架 IFrame

浮动框架 IFrame 是一种特殊的框架，它允许在浏览器窗口中嵌套子窗口，在其中显示子页的内容。例如，假设用户要制作一个如图 4-31 所示的页面布局。

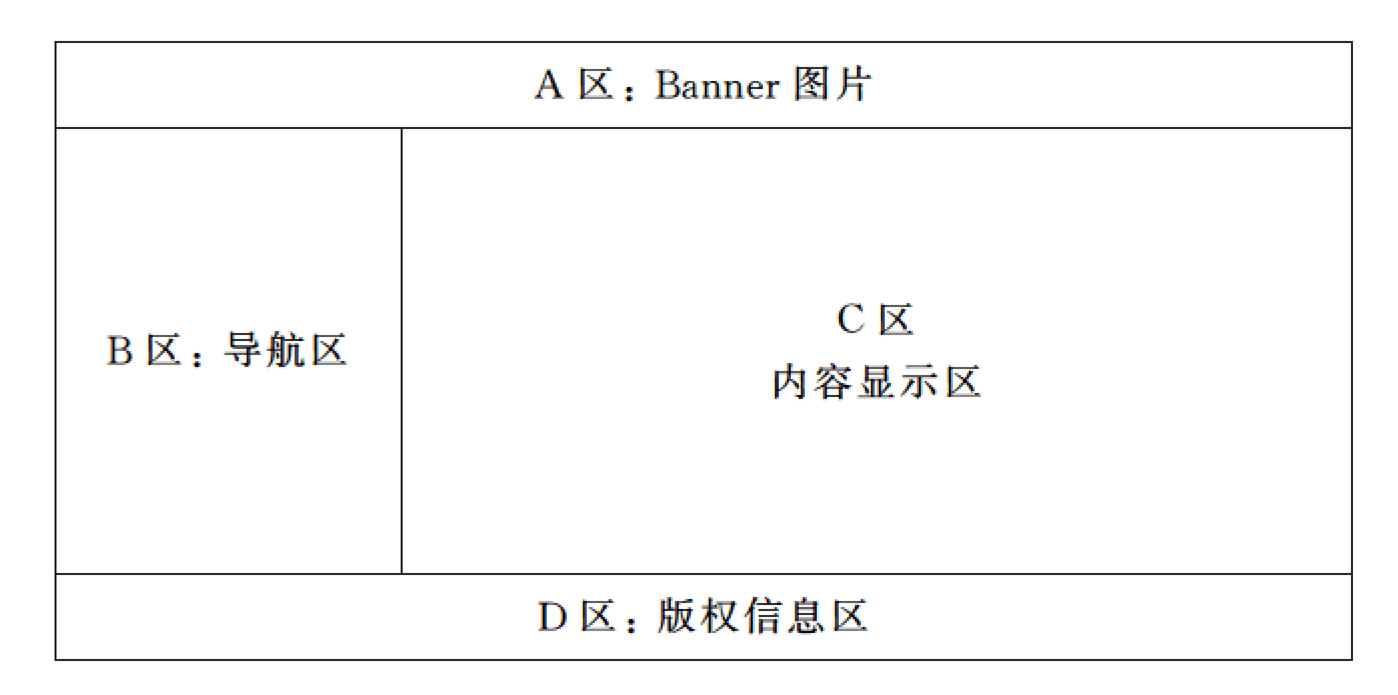



图 4-31　页面布局设计

如何来设计这个布局呢？如果用 Frameset 来设计，则需要设计四个 frame，如果 B 区和 C 区的内容较多时，会在相应的位置显示滚动条。但是，D 区的版权信息一直在屏幕上显示，占用屏幕空间，我们希望整个页面窗口出现滚动条，即，如果 B 区或 C 区的内容较多时，会出现窗口垂直滚动条，即 D 区需要拖动滚动条才能显示。

要实现上述要求，使用 Frameset 是不能实现的。应该通过 table 来布局实现，即设计一个 3×2 的表格，分别将第一行和第三行的两个表格合并，则得到上面的布局结构。当 B 区和 C 区的内容较多时，则在页面窗口自动出现垂直滚动条，D 区被推到后面不显示。

如果希望 B 区和 C 区的内容高度协调，应该在 C 区使用 IFrame，这样，当单击 B 区的一个超链接时，对应的页面在 C 区显示，如果页面内容超过 C 区的设定高度，则 C 区出现滚动条，示例代码清单如下：

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<style type="text/css">
  A:hover { color:#FF0000;text-decoration:none}
  A{color:rgb(235,239,71);text-decoration:none; font-size:12px}
.menutitle {cursor:hand;margin-left:8px; margin-top: 6px; margin-bottom:
           0px;background-color:rgb(0,119,166);
           color:rgb(254,254,166);width:150px;padding:8px; text-align:center;
           font-size:13px;height:30;border-right:1px solid #204848;
           border-bottom:1px solid #204848;border-left:1px solid #e0e0e0;border-top:
1px solid #e0e0e0;}
.submenu{margin-top: 0px;margin-bottom: 6px;margin-left:8px; background-color:rgb(0,
177,234);
          color:rgb(235,239,71);width:150px;padding:10px; text-align:center;line-height:
200%;
          font-size:13px;display:none;border-right:1px solid #204848;
          border-bottom:1px solid #204848;border-left:1px solid #f0f0f0;border-top:
1px solid #f0f0f0;}
</style>
<script>
function SwitchMenu(obj)
{
   if(document.getElementById)
   {
         var el = document.getElementById(obj);
         var ar = document.getElementById("masterdiv").getElementsByTagName("span");
         if (el.style.display != "block")
         {   for (var i=0; i<ar.length; i++)
             {
               if (ar[i].className=="submenu")
                   ar[i].style.display = "none";
              }
             el.style.display = "block";
         } else el.style.display = "none";
```

```
    }
}
</script>
</head>
<body topmargin="10">
<div align="center">
<table border="0" width="900" cellspacing="0" cellpadding="0" bgcolor="#FFFFFF"
height="248">
      <tr> <td width="900" colspan="2" height="90"> </td> </tr>
      <tr>
      <td width="905" height="39" colspan="2" background="image/bgpic001.gif">
        <marquee width="660" height="16">最新消息…</marquee>
      </td>
      </tr>
      <tr>
      <td width="165" background="image/bgpic004.gif" valign="top" height="154">
        <div id="masterdiv" align="left">
        <div class="menutitle" onclick="SwitchMenu('sub01')">教学内容</div>
        <span class="submenu" id="sub01">
             <a href="jpkc/kcjj.htm" target="tt">课程简介</a><br>
            <a href="jpkc/jxdg.htm" target="tt">教学大纲</a><br>
        </span>
        <div class="menutitle" onclick="SwitchMenu('sub02')">教学队伍</div>
        <span class="submenu" id="sub02">
             <a href="jpkc/hao.htm" target="tt">课程负责人</a><br>
             <a href="jpkc/zjjs.htm" target="tt">教学团队</a><br>
        </span>
……
……
        </div>
      </td>
      <td width="733" height="617" background="image/bgpic003.gif" align="center"
          style="border-right:1 solid #000040; border-bottom:1 solid #000040">
          <iframe name="tt" src="jpkc/kcjj.htm" width="728" height="659" scrolling=
"yes" frameborder="0">
          </iframe>
        </td>
      </tr>
      <tr>
      <td width="900" height="36" colspan="2" align="center">版权所有：…</td>
      </tr>
</table>
</div>
</body>
</html>
```

根据上述代码完成的网页示例如图 4-32 所示。




图 4-32　包含 IFrame 的网页示例

在上述界面中，左侧为折叠式菜单，右侧为内部帧，单击左侧的菜单项目，对应的页面将在右侧帧中显示。

# 思　考　题

1. 什么是 MVC 设计模式？按照 MVC 设计模式，网页功能可以分为哪些类型？
2. 请上网进行搜索，总结一下有哪些常用的网页布局。
3. 什么是页面视觉设计？有哪些主要的内容？
4. 简要说明页面设计的主要步骤。在页面效果设计中，为什么要进行切图？
5. 要对 Windows 平台中的网站进行远程管理，应如何设置？
6. 如何使用 FrontPage 进行远程网站的维护？
7. 怎样在 HTML 文档中插入图片？<img>标记的常用属性有哪些？怎样将这幅图片放在居中位置？
8. 怎样设置文本或图片超链接？
9. FrontPage 中不同的显示模式各有什么用途？
10. 在“图片属性”对话框中可以进行哪些属性的设置？怎样将图片设置成文字环绕的形式？
11. 什么是图片的“热点”区域？怎样设置多边形的“热点”？
12. 在“表格属性”和“单元格属性”中，cellspacing 和 cellpading 分别代表什么意思？

# 第 5 章

# 客户端开发

网页不仅仅是由 HTML 标记构成的，通常还包含脚本程序，它们增加了页面的交互和计算能力，使网页更生动、功能更强大。因此，网页设计除了必须掌握 HTML 的各种标记外，还应该熟悉脚本语言编程。脚本语言编程包括两个方面，一方面是客户端编程，另一方面是 Web 服务器端编程，它们分别在浏览器中和 Web 服务器上执行。本章介绍客户端的编程问题，关于服务器端的编程将在下一章介绍。

## 5.1 浏览器与客户端脚本程序

客户端脚本程序是指在客户的浏览器中运行的程序，这种程序不需要事先编译，如果浏览器从服务器上下载的网页中包含客户端脚本程序，浏览器将对脚本程序代码进行解释执行，即：浏览器通过自带的脚本引擎对 HTML 文档中的脚本程序进行分析、识别、解释并执行。

### 5.1.1 浏览器与客户端脚本引擎

浏览器之所以能够解释执行网页中的客户端脚本程序，是因为浏览器中内置了脚本引擎模块，可以解释执行网页中的客户端脚本程序。客户端脚本程序通常是用脚本程序语言书写的，脚本程序语言和传统的编译型程序设计语言（如 C++、Java 等）相比，在语法结构上类似，最大的不同是脚本程序设计语言编写的程序不需要编译链接过程，即不生成在操作系统下运行的 exe 文件，而是直接在浏览器中被浏览器解释执行。在解释执行过程中，如果程序存在错误，浏览器将停止程序的执行，并在浏览器窗口的状态栏中显示“网页存在错误”的提示。

由于安全方面的原因，浏览器可能禁止脚本程序的运行。例如，在浏览器的“Internet 选项”对话框中，选择“安全”选项卡，选择“Internet”，单击“自定义级别”，在安全设置列表中，可以在“活动脚本”中选择“禁用”、“启用”或“提示”。如果选择“禁用”，则浏览器将不执行网页中的客户端脚本程序。

### 5.1.2 脚本语言规范与主要的客户端脚本语言

1995 年末,网景公司(Netscape)发布了其客户浏览器脚本程序语言 JavaScript。随后,Netscape 将 JavaScript 语言规范提交给欧洲计算机制造协会(European Computer Manufacturing Association,ECMA)进行审定。1997 年 6 月,ECMA 发布了名为 ECMAScript Edition 1 的脚本语言规范,又称 ECMA-262。ECMAScript 不与任何具体浏览器相绑定,在浏览器之外,ECMAScript 描述了有关脚本程序语言所具有的通用属性,包括:语法、类型、语句、关键字、保留字、运算符、对象等,这些内容需要由具体的浏览器实现。

1998 年,微软在 IE 4.0 中发布了 JScript 3.0,宣称成为第一个遵循 ECMA 规范来实现的 JavaScript 脚本引擎。一年后,Netscape 发布其符合 ECMA 规范的 JavaScript 1.3。1999 年 12 月,ECMAScript 第三版发布,该版本成为脚本程序语言所遵循的标准,目前的所有脚本语言都可以看做是 ECMAScript 规范的一种具体实现。例如 JavaScript 就是由三个部分组成的,即:ECMAScript 规范、文档对象模型 DOM 和浏览器对象模型 BOM。

主要的客户端脚本语言有 JavaScript、JScript 和 VBScript 等,其中,JavaScript 是最早的客户端脚本语言,它是由 Netscape 随着其 Navigator 浏览器一起推出的,是浏览器默认的脚本程序语言。后来微软开发了 IE 浏览器,为了在 IE 中支持 JavaScript 脚本程序,微软开发了一个类似的脚本语言,即 JScript。可以说,在 IE 浏览器中,JavaScript 和 JScript 是等价的,两者在语言上几乎没有什么区别,但是两者提供的对象差别较大。

在网页中书写脚本程序,脚本程序应该书写在<script>…</script>标记对内。在网页中包含脚本程序的一般形式是:

```
<script language = " " runat =  " " type = "">
      语句部分
</script>
```

其中,language 参数用于设定脚本程序语言,取值可以是 JavaScript、Jscript、VBScript 等,如果是 JavaScript,可以省略不写。另外,在 language 参数中,还可以指定脚本语言的版本号,例如:language="Javascript1.3"。runat 参数设定脚本是否在服务端运行,如果设置脚本为服务端脚本,可以设置 runat= "server"。参数 type 指定媒体类型,例如,type="text/javascript"设置为文本文件类型。

脚本程序可以出现在网页的头部,也可以出现在网页的文档体中,其中出现在文档头部的脚本程序通常是一些函数,这些函数只有在显式地被调用时才被执行。在介绍具体的客户端脚本程序设计以前,先看两个包含 JavaScript 客户端脚本程序的简单网页示例。

**【例 5-1】** 检测浏览器中的 JavaScript 脚本程序版本(exa5-1.htm)。

```
<html>
<head>
<title>JavaScript 检测</title>
</head>
<body>
<h1>JavaScript 检测</h1><hr>
<!--JavaScript 支持性检测-->
```

```
<script type = "text/javascript">
        document.write("浏览器支持 JavaScript! <br><br>");
</script>
<!--JavaScript 版本检测-->
<script language = "JavaScript1.0">
        document.write("浏览器支持 JavaScript 1.0<br>");
</script>
<script language  = "JavaScript1.1">
        document.write("浏览器支持 JavaScript 1.1<br>");
</script>
<script language = "JavaScript1.2">
        document.write("浏览器支持 JavaScript 1.2<br>");
</script>
<script language = "JavaScript1.3">
        document.write("浏览器支持 JavaScript 1.3<br>");
</script>
<script language = "JavaScript1.4">
        document.write("浏览器支持 JavaScript 1.4<br>");
</script>
<script language = "JavaScript1.5">
        document.write("浏览器支持 JavaScript 1.5<br>");
</script>
<!--如果浏览器不支持 JavaScript,浏览器会显示相应的信息-->
<noscript>
        浏览器当前设置不支持 JavaScript!
</noscript>
</body>
</html>
```

分别使用 IE 6.0、Maxthon 2(遨游)和 Mozilla Firefox 2.0 打开上述网页文件 exa5-1.htm，在 IE 6.0 和 Maxthon 2(遨游)浏览器中，显示浏览器支持 JavaScript 1.1、1.2 和 1.3，因为 Maxthon 浏览器采用了 IE 内核，因此，两者对于 JavaScript 版本的支持一致。在 Mozilla Firefox 2.0 浏览器中，显示浏览器支持 JavaScript 1.0 到 JavaScript 1.5 的全部 JavaScript 版本。

和所有的软件开发一样，Web 页面中的程序也需要对程序进行结构化设计。在 JavaScript 中，同样提供了函数定义与函数调用功能，以支持结构化程序设计。由于浏览器浏览的 Web 页是顺序地从 Web 服务器调出，并由浏览器解释执行的，函数必须遵循先定义(一般放在<head>…</head>)后调用(在<body>…</body>内)的原则。

**【例 5-2】** 一个包含 JavaScript 函数的实例(exa5-2.htm)。

在下面的例子中，在页面的头部定义了一个计算正整数阶乘的函数，该函数将在文档体中被调用。

```
<html>
<title>Function in JavaScript </title>
<head>
<script language = "Javascript">
function fact(n)
{
```

```
    if (n==0)
       return 1;
    else
       return n * fact(n - 1);
  }
</script>
</head>
<body>
<p>fact(5) =
<script language = "JavaScript">
   document.write(fact(5));
</script>
</body>
</html>
```

在浏览器中打开上述网页文件,在浏览器中显示:fact(5)=120。

JavaScript 程序和一般的编译型程序不同,它没有一个操作系统调用的主程序(如:C 语言中的 main()函数),一个 JavaScript 函数定义时并不发生作用,只有在引用时(函数定义后的 document.write 语句)才被激活。

因为不同的浏览器和浏览器版本支持的脚本语言版本也不相同,因此,在书写客户端脚本程序时,应该根据浏览器的种类和版本,使用合适的内置对象和浏览器对象,否则,如果程序中使用了一些高版本脚本语言包含的对象,则网页在不支持该版本的浏览器中或低版本的浏览器中可能不能正常的显示。

## 5.2 JavaScript 程序设计基础

JavaScript 是一种基于对象的、事件驱动的浏览器脚本编程语言。1995 年,Netscape 开始研发一种开发代号为"魔卡(Mocha)"的脚本语言,根据 Netscape 的 Live 战略,该语言很快更名为 LiveScript,到了 1995 年末,为了迎合市场对 Java 语言的热情和市场营销的需要,该脚本语言正式地命名为 JavaScript。由于名称的缘故,容易使人们将 JavaScript 和 Java 联系在一起,但是,无论从思想和应用环境上,JavaScript 和 Java 语言都有着本质的区别。

### 5.2.1 JavaScript 基本符号

任何一种程序设计语言都有其自身的字符集和基本符号,它们按照语法构成程序语句,然后语句再构成程序。

#### 1. 基本字符

JavaScript 语言的基本字符有字母(a、b、…、z,A、B、…、Z)、数字(0、1、…、9)和特殊符号(+、-、*、/、<、=、>等)三大类。

同 C 语言一样,JavaScript 中同样有些以反斜杠(\)开头来表示不可显示的特殊字符,通常称为控制字符。

### 2. 关键字

由字母构成的具有固定含义的单词，如 var 代表变量说明，if 表示条件语句等。JavaScript 的关键字很多，可以参考 JavaScript 的专门书籍。

需要特别注意的是，JavaScript 是识别大小写的，在 JavaScript 中，关键字需要小写，如果书写有误，将显示“JavaScript 脚本错误”的警告。

### 3. 标识符

表示常量、变量和函数等名称的符号。标识符分为标准标识符和用户自定义标识符。标准标识符是表示标准常量和标准函数等名称的符号。JavaScript 中的标准常量和标准函数见下面的介绍。

用户自定义标识符是指用于说明常量、变量和函数等名称的符号。用户自定义标识符必须以字母开始，由任意的字母、数字和“_”组成。用户在命名标识符时应该有一定的命名规范：首先，用户自定义标识符不应该与标准标识符重名。第二，用户自定义标识符要尽可能反映它所代表的对象的含义。如：用 Name 代表名称比用 x 代表名称可读性更强。第三，如果是一个变量名，还应该尽可能反映变量的数据类型和作用域，比如用 nUserID 表示一个用户标识，最前面的小写字母 n 代表变量为整数类型。第四，大小写混写和较长的标识符可以把含义表达得更清晰。总之，好的命名规范可以提高程序的可读性，便于程序的维护。

### 4. 注释

为了增加程序的可读性，一般在程序中增加注释语句，注释内容没有语法要求，其内容仅仅是一个提示作用。在 JavaScript 中，注释以“//”字符引导，注释可以单独一行，也可以在语句行的后面。

需要说明的是，与 HTML 的注释（<!-- .... -->）不同，当在服务器端处理脚本时，JavaScript 中注释将被删除，而不是被送到浏览器。若需要客户端浏览器看到脚本中的注释，应该使用 HTML 注释将注释加进 HTML 页。此时，注释将返回给浏览器。

## 5.2.2 数据和数据类型

JavaScript 脚本程序语言是一种解释性的程序语言，所写的脚本程序不需要编译和链接，它采用边读边执行的方式运行。在数据类型方面，JavaScript 提供了四种基本的数据类型用来处理数字和文本。

JavaScript 提供的数据类型有：数值（整数和实数）、字符串型（用双引号"或单引号'括起来的字符）、布尔型（True 和 False）。

在 JavaScript 的基本类型中的数据可以是常量，也可以是变量。由于 JavaScript 采用弱类型的形式，因此变量不必事先声明其数据类型，可以在使用或赋值时确定其数据类型。此外，一个变量的类型在使用时还可以被改变。

通常情况下，可以先给变量赋一个初值，通过初值的类型来声明该变量的数据类型，这

更加符合一般的程序设计思想,例如:

```
var x = "hello";
```

上述语句声明了一个变量 x,通过为 x 赋初值"hello",将 x 作为一个字符串类型变量。在后面的语句中,可以给 x 赋一个整数,如:x=100,这样变量 x 就成为整数类型了。在 JavaScript 中,上述变量类型的改变不会和其他程序设计语言一样出现赋值不相容的错误。

## 5.2.3 常量和变量

### 1. 常量和常量定义

常量是指在程序执行过程中,其值不发生变化的量。常量有字面常量和符号常量两种。字面常量就是一些数值或字符串,例如:3.14、"hello"等都是字面常量。根据数据类型的不同,常量可分为整数常量、实数常量、字符常量等。字符型常量是指使用单引号(')或双引号('')括起来的一个或多个字符。如'a'、"hello"、"5123"、"x+y" 等,其中单引号括起来的为单个字符,双引号括起来的为字符串。符号常量是指为一个常量起一个名字,即常量名,常量名是一个用户自定义标识符。例如:用 pi 代表圆周率 3.14。

常量命名有两方面的好处,首先恰当的常量名称可以增加程序的可读性;其次,使用常量名可以便于程序的维护。例如,可以命名常量 pi=3.14,如果希望提高求解精度,可以修改 pi 的定义为 pi=3.1416,这种修改只在一处进行,不会产生不一致的情况,而且修改简单。

### 2. 变量和变量说明

所谓"变量"是指在程序执行过程中,其值发生变化的量。每一个变量都有一个变量名,对应一个特定的内存空间。变量有两个重要的属性,一个是数据类型,一个为操作运算。数据类型决定了变量所占内存空间的大小,也决定了数据的取值范围和操作运算。例如,一个整数类型的变量在内存中占两个字节,能够进行算术运算和关系运算等。字符类型变量占一个字节,能进行关系运算,但不能进行算术运算。

在 JavaScript 中,变量命名的一般形式是:

```
var <变量名表>;
```

其中,var 是 JavaScript 的保留字,表明接下来是变量说明,变量名表是用户自定义标识符,变量之间用逗号分开。和 C++等程序设计语言不同,在 JavaScript 中,变量说明不需要给出变量的数据类型。此外,变量也可以不说明而直接使用。

例如:var x,y;则定义了两个变量,名称分别为 x 和 y,没有给出具体的数据类型,也没有赋予它的值。

再如:var myName=''John '',则定义了一个变量 myName,同时赋予了它一个字符串值。在 JavaScript 中,变量可以不作声明,在使用时,数据的类型将确定变量的类型。

### 3. 变量的作用域

变量的作用域由声明变量的位置决定,决定哪些脚本命令可访问该变量。在函数外部声明的变量称为全局变量,其值能被所在 HTML 文件中的任何脚本命令访问和修改。在

函数内部声明的变量称为局部变量。只有当函数被执行时，变量被分配临时空间，函数结束后，变量所占据的空间被释放。局部变量只能被函数内部的语句访问，只对该函数是可见的，而在函数外部则是不可见的。

### 5.2.4　表达式和运算符

表达式是指将常量、变量、函数、运算符和括号连接而成的式子。根据运算结果的不同，表达式可分为算术表达式（结果为整数或实数）、字符表达式（结果为字符或字符串）和逻辑表达式（结果为 true 或 false）。

JavaScript 提供了丰富的运算功能，包括算术运算、关系运算、逻辑运算和连接运算等。

#### 1. 算术运算符

JavaScript 中的算术运算符有单目运算符和双目运算符。双目运算符包括：＋（加）、－（减）、*（乘）、/（除）、%（取模）、|（按位或）、&（按位与）、<<（左移）、>>（右移）等。单目运算符有：－（取反）、～（取补）、＋＋（递加 1）、－－（递减 1）等。

#### 2. 关系运算符

关系运算又称比较运算，运算符包括：<（小于）、<＝（小于等于）、>（大于）、>＝（大于等于）、＝（等于）和！＝（不等于）。

关系运算的运算结果为布尔值，如果条件成立，则结果为 true，否则为 false。

#### 3. 逻辑运算符

逻辑运算符有：&（逻辑与）、|（逻辑或）、！（取反，逻辑非）、^（逻辑异或）。

#### 4. 字符串连接运算符

连接运算用于字符串操作，运算符有＋（用于强制连接），将两个或多个字符串连接为一个字符串。

#### 5. 三目操作符

三目操作符“?：”格式为：

```
操作数？表达式 1：表达式 2
```

三目操作符“?：”构成的表达式，其逻辑功能为：若操作数的结果为 true，则表达式的结果为表达式 1，否则为表达式 2。例如 max＝(a>b)？a：b，该语句的功能就是将 a、b 中的较大的数赋给 max。

### 5.2.5　基本语句

所有的程序设计语言，无论是过程式程序设计语言还是面向对象的程序设计语言，其程序设计语句都可以分为三种类型，即：顺序语句、分支语句和重复语句，使用这三种语句就

可以描述用户的所有业务逻辑。

### 1. 顺序语句

从本质上讲,在一个程序中,语句的执行总是从上而下顺序执行的。在过程式的程序设计语言中,这种顺序是程序本身所显式地定义的。例如 C 语言中的函数调用,遇到函数调用,转去执行相应的函数,执行结束后返回。在面向对象的程序设计语言中,函数的调用变得更加复杂,它是在程序的运行过程中,由事件触发消息,由消息来激活函数。这样的事件驱动机制,使函数的调用变得不再像过程式程序设计中的显式调用那么清晰,但这种消息影射(message map)机制降低了函数之间的耦合度,大大增强了软件系统的可维护性,在函数内部,语句仍然是从上到下顺序执行的。

无论是 C、C++、Java,还是 JavaScript,由于语句语法的需要,有时候需要将多个语句看作逻辑上的一个语句,此时需要将这多个语句之间用分号“;”分开,然后用一对花括号“{”和“}”括起来,称为语句块。在语句块内部,语句从上而下顺序执行。

### 2. 分支结构

在 JavaScript 中,实现分支结构的语句有三种,它们是条件判断语句 if…语句、if…else…语句和开关语句 switch 语句。

(1) if 语句

if 语句的一般形式是:

```
if (<条件表达式>)
    <语句>;
```

if 语句首先计算条件表达式的值,若计算结果为 true,则执行语句部分,否则执行 if 语句下面的语句。if 语句逻辑功能如图 5-1 所示。

语句部分逻辑上是一个语句,如果语句部分需要多个语句来实现,应将这多个语句用“{”和“}”括起来,形成语句块,作为一个逻辑上的语句。

```
例如: if (a>=b)
      {
        max=a;
        min=b;
      }
```

(2) if…else 语句

if…else 语句先计算条件表达式的值,根据计算结果确定要运行的语句。一般形式是:

```
if (<条件表达式>)
   <语句 1>;
else
   <语句 2>;
```

if…else 语句首先计算条件表达式的值,若为 true,则执行语句 1,否则,执行语句 2,逻辑功能如图 5-2 所示。

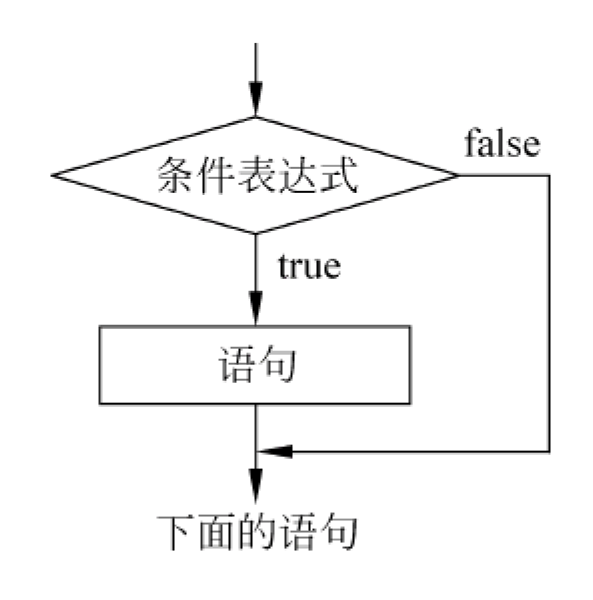



图 5-1 if…语句逻辑功能图

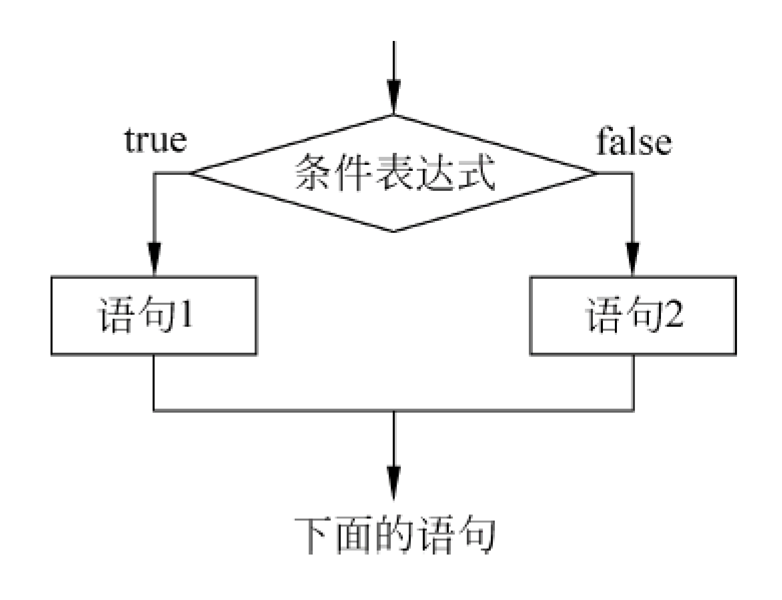



图 5-2 if…else 语句逻辑功能图

在存在两种分支的情况下，if…else 语句可以很好地描述这种逻辑。在多种分支的情况下，可以使用 if…else 语句的嵌套进行描述，但是嵌套降低了程序的可读性。为此，对于多种分支的情况，JavaScript 提供了 switch 语句。

（3）switch 语句

switch 语句提供了 if…else 多层嵌套结构的一个变通形式，可以从多个语句块中选择执行其中的一个。switch 语句提供的功能与 if…else 语句类似，但是可以使代码更加简练易读。switch 语句的一般形式为：

```
switch (<数值或字符串表达式>)
{
      case 表达式 1
           语句 1;
           [break;]
      case 表达式 2
           语句 2;
           [break;]
           …
      case 表达式 n
           语句 n;
           [break;]
   [default:
           语句 n+1;
   ]
}
```

在 switch 语句的开始是一个条件表达式，该条件表达式可以是一个整数、实数或字符串表达式。表达式的结果将与结构中每个 case 的值比较。如果匹配，则执行与该 case 关联的语句块。执行完后，如果希望退出 switch 语句，则需要使用 break 语句，否则，将继续下面的 case 匹配。switch 语句的逻辑功能如图 5-3 所示。

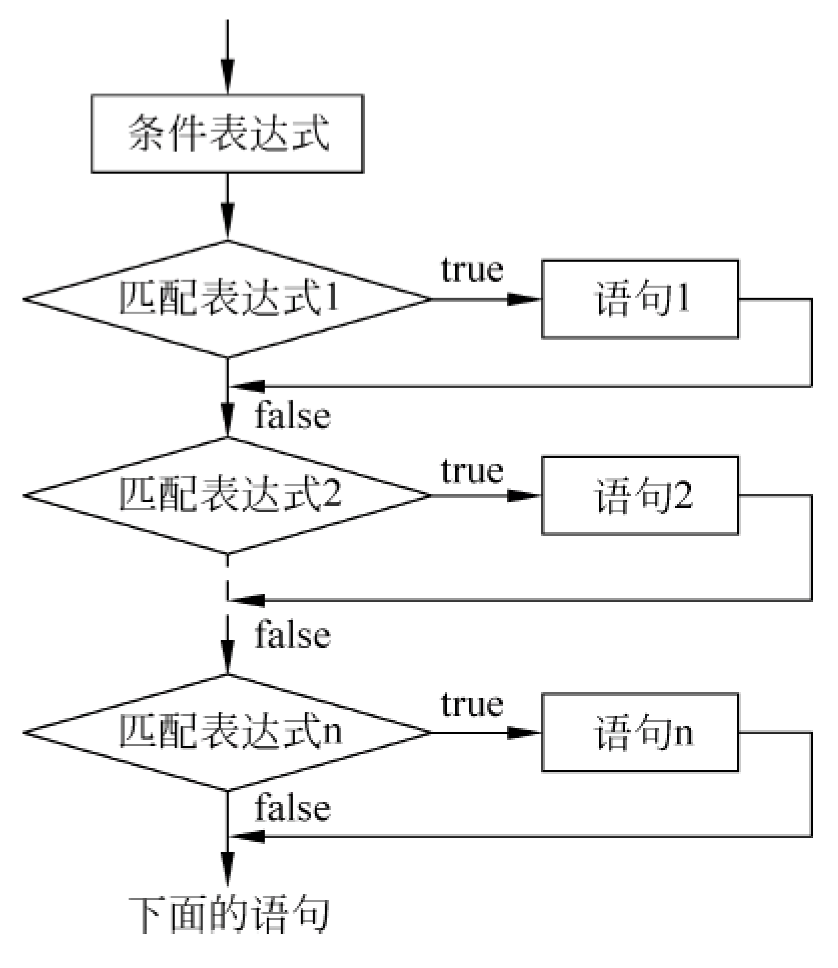



图 5-3 switch 语句逻辑功能图

### 3. 重复结构

当部分语句需要反复执行时，需要使用重复语句。重复语句总是由循环体和循环终止

条件两部分组成,在 JavaScript 中可使用下列两种形式的重复语句。

(1) while 循环语句

while 循环语句是先判断循环终止条件,然后再执行循环体,因此循环体可能一次也不被执行。while 循环语句的一般形式是:

```
while (<条件表达式>)
    <语句>;
```

首先计算条件表达式的值,若计算结果为 true,则执行循环体语句,然后无条件地返回 while 语句开始,继续计算条件表达式的值。如果条件表达式计算的结果为 false,则结束循环,执行 while 循环语句下面的语句。逻辑功能如图 5-4 所示。

(2) for 循环语句

一般形式为:

```
for (表达式 1; 表达式 2; 表达式 3)
   <语句>;
```

执行过程如下:

Step1:计算表达式 1。

Step2:计算表达式 2,如果结果为 true,则执行循环体,然后转 Step3。否则,结束循环。

Step3:计算表达式 3。

Step4:无条件转 Step2。

Step5:循环结束,执行 for 语句下面的语句。

逻辑功能如图 5-5 所示。

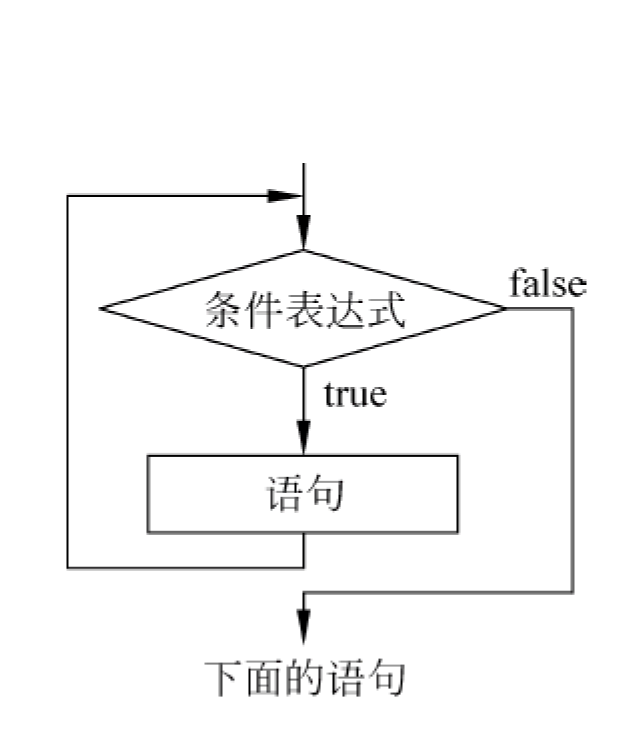



图 5-4 while 语句逻辑功能图

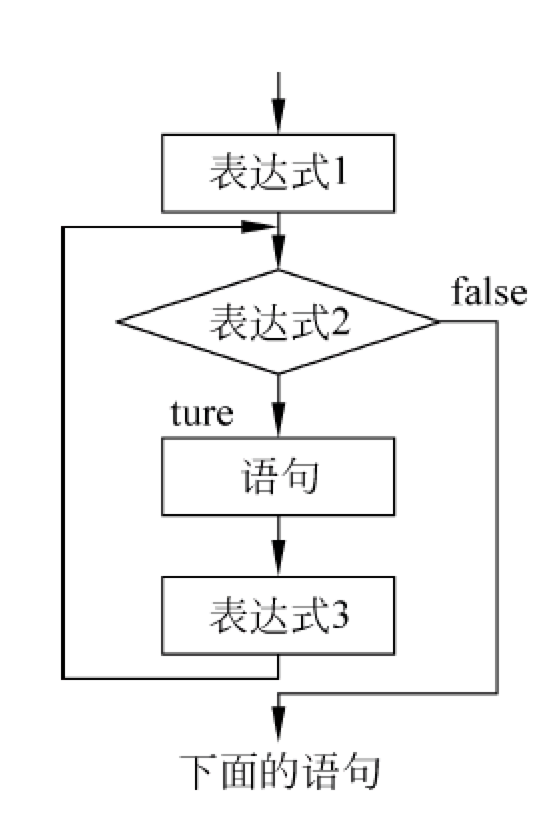



图 5-5 for 语句逻辑功能图

在 for 语句中,表达式 1、表达式 2 和表达式 3 中可以省略其中的一个或多个,但它们之间的分号不能省略。例如,可以写下面的 for 语句形式:

```
for (; ;)
{
…
}
```

上述 for 语句是一个死循环，在循环体中必须包含 break 语句，以结束循环的执行。另外，循环体的内部也可以包含循环语句，即循环的嵌套，构成多重循环。需要特别注意关键字要小写，例如 for 不应该写成 For。

#### 4. break 和 continue 语句

与 C++ 语言相同，使用 break 语句使得程序执行从 for 语句或 while 语句中跳出，continue 使得跳过循环内剩余的语句而进入下一次循环。

### 5.2.6　函数

在结构化程序设计中，函数是实现结构化程序设计的主要手段，它把一个系统中需要反复多次执行的部分定义为一个函数。如判断一个数是否为素数、求两个数的最大公约数等。在 JavaScript 语言中，已经给出了许多标准函数，同时也允许用户自己定义函数。

在 JavaScript 中，函数是以 function 开头定义的，由函数头部和函数体构成，一般形式为：

```
function <函数名> (<形式参数表>)
{
    变量说明部分;
    语句(函数体);
}
```

在函数的定义中，函数名后有形式参数表，这些形式参数变量可能是一个或几个，在 JavaScript 中可通过函数名. arguments . Length 来检查参数的个数。在函数体中，可以分为变量说明和语句部分，变量说明用于说明该函数所要处理的数据，语句部分是对变量的处理。在 JavaScript 中，由于数据类型采用弱数据类型，变量说明部分可以省略不写。在函数体的语句部分必须要包含一个返回函数值的语句，即 return 语句。

## 5.3　事件驱动及事件处理

JavaScript 是一种基于对象（Object Based）的程序设计语言。和面向对象（Object Orinted）的程序设计语言不同，它没有对象的继承、多态等机制。但是，在 JavaScript 中，包含了许多内置对象，可以分享面向对象的程序设计的优势，同时，也具有面向对象程序执行的特征。

### 5.3.1　事件驱动的程序执行过程

在面向对象的程序设计中，事件、消息和事件处理程序是程序运行过程中的核心概念。所谓事件（Event），一般是指用户在某个操作对象上所执行的键盘或鼠标操作。当某个事件发生时，将自动产生一系列的消息（Message），每个消息对应一个消息处理函数（Event Handler），从而生成一个消息影射（Message Map）。每一个程序有一个消息循环，用户的操

作通过消息影射来调用和执行相应的函数代码。消息影射使得函数之间做到了更大程度的解耦,从而大大地增强了程序的可维护性和可扩展性。

在 Web 中,浏览器的 JavaScript 脚本引擎对于页面文档,在内存中创建一个 DOM 树,对于网页中的每一个标记,在 DOM 树上对应一个结点。对于每一个标记,可以设置其事件属性,在事件属性中给出对应的事件处理函数,这些函数实现对 DOM 的操作,即对网页内容进行操作,这使得原先的静态页面成为可编程的。

### 5.3.2 JavaScript 中的常用事件

在 JavaScript 中,对于不同的页面元素,定义了相应的事件。用户可以设置这些标记的事件属性,从而来定义相应的行为,实现对页面的操作。不同的页面元素,其可实施的事件属性不同,这些属性可以通过 FrontPage、Dreamweaver 等可视化的页面制作工具来查询。在 JavaScript 中,鼠标和键盘操作事件及可操作的元素对象见表 5-1。

**表 5-1 页面元素与对应事件列表**

| 动 作 | 事 件 | 动 作 对 象 |
|---|---|---|
| 单击鼠标 | onclick | <body>、<p>、<img>、<a>、<form>以及表单中的各种表单域元素 |
| 双击鼠标 | ondblclick | 同上 |
| 鼠标置于元素对象之上 | onmouseover | 同上 |
| 鼠标按下 | onmousedown | 同上 |
| 鼠标抬起 | onmouseup | 同上 |
| 鼠标移动 | onmousemove | 同上 |
| 鼠标离开元素对象 | onmouseout | 同上 |
| 键被按下 | onkeydown | 同上。按键的 ASCII 码值保存在 window. event. keyCode 中 |
| 键被按下然后被释放 | onkeypress | 同上 |
| 键被释放 | onkeyup | 同上 |
| 改变浏览器窗口大小 | onresize | <body> |
| 滚动元素对象 | onscroll | <body>、<img> |
| 元素对象获得焦点 | onfocus | <body>、<img>、<a>、各<form>域元素 |
| 元素对象失去焦点事件 | onblur | <body>、<a>、各<form>域元素 |
| 选中事件 | onselect | 当<form>中的 Text 或 Textarea 对象中的文字被加亮后,引发该事件 |
| 改变元素对象值 | onchange | 当利用<form>中的 text 或 texturea 元素进行输入,内容改变时发出该事件,同时当在 select 表格项中一个选项状态改变后也会引发该事件 |

不同的元素对象,其对应的事件也不相同,在 FrontPage 中,在"代码"视图下,利用 IntelliSense 技术可以看到每一个元素的所有可能事件。

对于元素的每一个事件,可以设置相应的事件处理函数,例如:

```
<form>
    <input type = "button" name =  "B1" value = "test" onclick = "change()">
```

```
    <input type = "text" name = "MyAddress" value = "RoadNo, City"
           onchange = "check(this.value)">
</form>
```

在上述代码中，在 button 按钮上单击将激活 change()函数，修改 text 文本框的内容时将激活 check()函数。对于 button 按钮的 onclick 属性和 text 的 onchange 属性，可以使用用户编写的函数作为事件处理程序，也可以使用 JavaScript 中的内部函数，也可以直接使用 JavaScript 代码等。例如：

```
<a href = "#" onclick = "window.close()" >关闭窗口</a>;
```

上述代码是许多页面中见到的，“关闭窗口”超链接为一个空超链接，当用户点击“关闭窗口”超链接时，当前窗口被关闭。

## 5.4　对象及其操作

在软件系统的开发中，面向对象的程序设计是目前最主流的程序设计思想，大部分的软件开发工具也是面向对象的，例如 C++、Java 等。面向对象的思想将问题域中的每一个实体都影射为软件系统中的一个对象，通过类和对象，实现了数据和数据操作的封装，极大地增强了系统的灵活性、可维护性和可扩展性。

### 5.4.1　类与对象的概念

类(Class)和对象(Object)是面向对象的程序设计(Object Oriented Programming，OOP)的灵魂。所谓类，它是指一种包含成员变量和成员函数的数据结构。类和传统的过程式程序设计中的数据类型相似，它不占用内存空间，主要目的是用于声明对象。对象(Object)是用类来声明的数据结构，如果将类比作数据类型，对象就是相应数据类型的变量，在内存中分配特定的空间，存储数据。

JavaScript 并不是一个完整的面向对象的程序设计语言，它不提供关于对象的抽象、封装及派生等功能。但它可以使用浏览器的内置对象，也可以允许用户自定义对象，从而分享面向对象程序设计带来的好处。

JavaScript 定义“类”和定义“函数”的语法是一样的，而且这样的函数称为该类的构造函数。用户用函数定义来定义类，然后用 new 语句创建该类的一个实例。

在 JavaScript 中，类定义的一般形式是：

```
function className(<属性表>)
{
    this.prop1 = prop1;              // 属性
    this.prop2 = prop2;
    …
    this.meth = FunctionName1;       // 方法,需要定义对应的函数
    this.meth = FunctionName2;
    …
}
```

对象属性和方法的引用主要使用点(.)运算符,一般形式为:

对象名.属性名|方法名

**【例 5-3】** 定义一个矩形类,包含三个成员变量,分别是矩形的高、宽和面积,另外包含一个计算面积的成员函数。

```
<html>
<head>
<script language="javascript">
// 类成员函数对应的函数定义
function compArea()
{
 return this.width * this.height;
}
// 定义一个矩形类,包含两个成员变量和一个成员函数
function MyClass(x,y)
{
   this.width = x;
   this.height = y;
   this.getarea = compArea;
}
</script>
</head>
<body>
<script language="javascript">
     var area=0;
     myObj = new MyClass(5,6);
     area=myObj.getarea();
     document.write(area);
</script>
</body>
</html>
```

在上述代码中,在页面的头部定义了一个矩形类 MyClass,包含两个成员变量 Width 和 Height,表示矩形的长和宽,定义了一个成员函数 getarea,用于计算并返回矩形的面积。在页面的<body>体内,声明了一个用户类的对象 myObj,计算并输出该矩形的面积。

### 5.4.2 对象的操作

在 JavaScript 中提供了几个用于操作对象的语句和关键字及运算符。

(1) for…in 语句

格式如下:for(对象属性名 in 已知对象名)

顺序输出对象各个属性的值,如果是方法,则输出对应的函数代码。例如:

```
function showData(object)
{
     for(var prop in object)
          document.write(object[prop]);
}
```

使用该函数时，在循环体中，可以不知道对象属性的个数，for 可以自动地将属性取出来，直到最后为止。

(2) with 语句

使用该语句，在该语句体内，任何对变量的引用被认为是这个对象的属性，从而节省程序代码。一般形式为：

```
with object{
…
}
```

(3) this 关键字

this 是对当前对象的引用，在 JavaScript 中，对象的引用是多层次的，往往一个对象的引用又需要对另一个对象的引用，而另一个对象有可能又要引用另一个对象，这样有可能造成混乱，为此 JavaScript 提供了一个用于将对象指定当前对象的指针 this。

(4) new 运算符

使用 new 运算符可以创建一个新的对象。创建对象的一般形式是：

```
myobj = new calssName(参数表);
```

其中，myobj 为创建的对象名称，calssName 是类的名称，参数表用于激活类的相应的构造函数，从而为类中的成员变量赋值。

如创建一个日期对象，代码为：

```
myDate = new Date();
myBirthday = new Date("January 18,1973 6:15:00");
```

这样就创建了两个新的日期对象 myDate 和 myBirthday，myDate 对象取当前的计算机时钟作为其日期值，而第二个 Date 对象 myBirthday 被赋值一个具体的日期。

## 5.5　常用内部对象及函数

每一种程序设计语言，都可以分成程序设计语法和标准库两个部分。语法部分通常用于定义一种程序语言的基本语法规范，包括：字符集、保留字、数据和数据类型、程序语句等内容。标准库则是指为用户提供的一组常用函数库或类库。对于结构化程序设计语言，例如 C 语言，标准库通常是一组标准的函数库，包含了大量的标准函数。如果是面向对象的程序设计语言，例如 C++、Java 等，标准库则是一组标准类库，包含了大量的标准类。使用标准函数库或类库，可以提高用户的编程效率和保证程序代码质量。

由于 JavaScript 是一种基于对象的脚本程序设计语言，它具有面向对象程序设计语言的特点，但又不完全是面向对象的程序设计。这体现在关于类库的设计方面，例如 JavaScript 不仅提供了一组标准类库，同时还提供了一组常用的对象和全局函数。而在面向对象的程序设计思想中，尽量不使用全局对象变量，所有的函数都应该隶属于一个类。不同的 JavaScript 版本，包含的内部对象和函数也不一样，下面介绍一些常用的内部对象和函数。

### 5.5.1 String 对象

在 JavaScript 中,每个字符串都是一个 String 对象。使用 String 对象时,不需要像一般自定义对象一样用 new 关键字在内存中创建对象,而是可以直接将一个字符串赋给一个变量。string 字符串对象封装了 JavaScript 中的字符串以及相关的操作。

字符串对象的生成十分简单,而且是隐式的,不使用 new 关键字,例如:

```
var myStr = "Hello";
```

这样,myStr 就是字符串对象了,一个变量被声明为字符串对象之后,它就拥有了这个对象类的属性和方法,可以和一般对象一样,使用对象的方法,取得对象的属性。

#### 1. 字符串对象的属性

字符串对象的属性只有一个,这就是 length(长度)属性,返回字符串的长度。需要说明的是,如果字符串为英文,那么长度属性的值是字母个数加上特殊符号个数,加上空格数;但是如果字符串是中文,每个中文单字占两个英文字符的长度。

例如:

```
<script>
  var myStr = "Hello";
  document.write(myStr.length);
</script>
```

上述代码在浏览器中输出字符串“Hello”的长度为 5。

#### 2. 查找字符和子串

在字符串类的成员函数中,有一类用于查找在字符串中某一特定字符的所在位置,或者是字符串中特定字符的有无,或者是特定位置的字符值。

(1) charAt()

这个方法返回 string 对象特定位置的字符值,一般形式是:

```
StrName.charAt(Position);
```

StrName 是字符串变量名或字符串常量。Position 用来指定希望取得的字符在字符串中的位置,它是大于等于 0,小于字符串长度属性(length)的正整数。例如:

```
var myStr = "abcdefg";
var FiveChar = myStr.charAt(4);
```

上述语句将取出字符串中的第 5 个字符,并将它存储于 FiveChar 变量中。需要注意,这个成员方法只适用于英文字符串,对于中文字符串,它将返回一个乱码,因此,要避免在中文字符串中使用这个成员方法。

(2) indexOf()

这个方法返回指定的字符(或字符串)在字符串中的位置,一般形式为:

```
StrName.indexOf(subStr); 或者
StrName.indexOf(subStr,StartPosition);
```

其中 StrName 是字符串对象名，可以是常量，也可以是字符串变量。subStr 是一个待查找的字符或者字符串，可以是常量，也可以是变量。在省略 StartPosition 的情况下，此函数将从字符串的第一个字符开始查找；当 StartPosition 参数存在的情况下，这个函数将从字符串中的第 StartPosition+1 个字符开始查找；当 StartPosition 超过字符串的长度时，返回−1。当所希望查找的字符串找不到时，返回−1 。当字符串中有两个以上的待查找字符串，则返回被搜寻字符串中位置在最前面的待查字符串的位置。

(3) lastIndexOf()

这个方法的使用与 indexOf()方法十分相似，indexOf()是按照从左到右的顺序查找待查找字符串，而 lastIndexOf()是按照从右到左的顺序查找待查找的字符串。

lastIndexOf()也可以指定开始查找的位置，指定的方法和 indexOf()一样。

(4) substring()

返回一个指定的子字符串，它的语法是：

```
StrName.substring(Position1,Position2);
StrName.substring(Position);
```

第一种格式中，返回下标从 Position1 到 Position2 之前的字符串(不包含下标是 Position2 的字符)。例如：substring(1,3)代表返回下标是 1～2 的字符，即字符串中的第二和第三个字符。这两个参数的大小关系可以任意，在调用时，取两参数中较小的值作为子串开始的下标值，取较大的参数减 1 作为子串结束的下标值。

第二种格式的 Position 用来指定子串开始的字符下标位置，而子串结束的位置取被搜索字符串的结尾。例如：substring(1)代表取第二个字符开始到结尾的所有字符。

#### 3. 改变字符串大小写

```
StrName.toLowerCase();
StrName.toUpperCase();
```

这两个成员方法分别将字符串中所有的字母变为小写字母或大写字母，将变化后的结果返回。但是，原字符串内的大小写不变。

例如：

```
<script>
   var myStr = "Hello";
   document.write(myStr.toUpperCase());
</script>
```

则输出大写的字符串“HELLO”。

#### 4. 形成 HTML 文本格式

网页中常常需要这样做，对一个字符串变量进行操作，使它变为一定的 HTML 特定格式。例如，一个字符串变量定义如下：

```
var aStr = "This is the Discount Table"
```

需要的格式是：

```
<a href = "home.html">This is the Discount Table</a>
```

可以利用字符串的"＋"操作来将其转化。但是 JavaScript 中提供了更简单的方法。

(1) fontcolor()方法

形成并返回一个指定颜色的字符串，它的语法格式是：

```
StrName.fontcolor(FontColor);
```

其中 FontColor 是一个颜色的名字，或者是一个用十六进制表示的颜色。例如：

```
var AStr = "Font Color Exmaple";
var FontColorText = AStr.fontcolor("green");
document.write(FontColorText);
```

那么显示的文字应是绿色的。相当于 FontColorText 具有下面的值：

```
<font color = "green">Font Color Example</font>
```

(2) fontsize()方法

形成并返回一个特定大小的字符串，一般形式是：

```
StrName.fontsize(FontSize);
```

其中 FontSize 参数是数值型的变量或者常数。现在 JavaScript 识别的字体有 7 种，FontSize 参数由 1～7，字体由小到大。当 FontSize 参数大于 7 时，将被当作 7 处理；当其小于 1 时，将被当作 1 处理；这个参数也可以是负数。当参数是－1 时有一点特殊，它相当于 FontSize＝2。

例如："FontSize Example".fontsize(4);

相当于：<font size = 4>"FontSize Example"</font>

(3) link()方法

用来形成并返回一个字符串，此字符串用于在网页中构造一个超链接。它的语法格式是：

```
StrName.link(Href);
```

其中 Href 是超链接的 URL。例如，下面的语句：

```
var AStr = "A Href Example";
var HrefText = AStr.link("home.html");
```

将使 HrefText 变量具有下面的值：

```
<a href = "home.html">A Href Example</a>
```

(4) anchor() 方法

这个方法用来形成并返回一个字符串，此字符串用于在网页中构造一个锚点。所谓"锚点"，是指当网页文档的内容比较多时，一般在这些网页中，可以通过单击一些超链接直接跳至此网页文档的一些特定位置，这些特定位置就是锚点。锚点有自己的名字，这和一个

URL 类似。当希望通过超链接到达锚点时，需要这个成员方法的语法格式为：

```
StrName.anchor(anchorName);
```

其中 anchorName 是与 HTML 中<a>标识中的 name(名字)属性相关的字符串。例如，我们这样使用：

```
var aStr = "This is the Discount Table ";
var anchorText  =  aStr.anchor("Discount");
document.write(anchorText);
```

则 anchorText 的取值是：

```
<a name = "Discount">This is the Discount Table </a>
```

于是，用 document.write()将这个字符串写入到网页中就有了一个名字是“Discount”的锚点，而这个锚点显示的文字是“This is the Discount Table”。

需要说明的是，JavaScript 是弱类型程序设计语言，在一个变量被声明为字符串变量之后，它仍旧可以被赋予数值，这时这个变量就不再是字符串了，而成为一个数值型的变量。例如：

```
var varStr = "name and address";
varStr =  1;
```

上述赋值语句使 varStr 成为一个数值型的变量，它不再拥有字符串的属性和方法。

## 5.5.2　Math 对象

JavaScript 中的 Math 对象封装了常用的数学常数和一些常用的数学运算，这些运算包括：三角函数、对数函数、指数函数和一些舍入函数等。

### 1. Math 对象的属性

Math 对象中的属性与其他对象的属性有一些区别。这些属性是常用的数学常数，它们是定值，因此它们是只读的，不允许对这些对象属性进行写操作。表 5-2 列出了 Math 对象的属性名、描述和近似值，以供参考。

**表 5-2　Math 对象的属性**

| 属性名 | 说　明 |
|---|---|
| E | 常数 e，或称做欧拉常数，它是自然对数的底。近似值为 2.718 |
| LN2 | 2 的自然对数，近似值为 0.693 |
| LN10 | 10 的自然对数，近似值为 2.302 |
| LOG2E | 以 2 为底的常数 E 的对数，近似值为 1.442 |
| LOG10E | 以 10 为底的常数 E 的对数，近似值为 0.434 |
| PI | π 常数，即圆周长和直径之比，近似值为 3.142 |
| SQRT1.2 | 0.5 的平方根，近似值为 0.707 |
| SQRT2 | 2 的平方根，近似值为 1.414 |

## 2. Math 对象的成员方法

(1) 三角函数

三角函数包括三角和反三角两类函数，Math 对象的三角函数见表 5-3。

表 5-3 三角函数和反三角函数

| 成员函数 | 说明 |
|---|---|
| acos(Value) | Value 的反余弦值，单位是弧度 |
| asin(Value) | Value 的反正弦值，单位是弧度 |
| atan(Value) | Value 的反正切值，单位是弧度 |
| atan2(Xvalue，Yvalue) | 直角坐标系中(Xvalue，Yvalue)点与 X 轴所成的角度 |
| cos(Value) | Value 的余弦值，Value 的单位是弧度 |
| sin(Value) | Value 的正弦值，Value 的单位是弧度 |
| tan(Value) | Value 的正切值，Value 的单位是弧度 |

(2) 对数、指数函数

这一类函数包括几个常用的对数和指数操作，另外，还包括常用的幂函数操作，见表 5-4。

表 5-4 对数和指数函数

| 成员函数 | 说明 |
|---|---|
| exp(Value) | E 的 Value 次方 |
| log(Value) | Value 的自然对数 |
| pow(BaseValue，ExpValu) | BaseValue 的 ExpValue 次方 |
| Sqrt(Value) | Value 的平方根 |

(3) 舍入函数

舍入函数的作用是将一些不合乎需要的数值转化为需要的数值，例如将浮点数转化为相应的整数，见表 5-5。

表 5-5 舍入函数

| 成员方法 | 说明 |
|---|---|
| abs(Value) | Value 的绝对值 |
| ceil(Value) | 大于或者等于 Value 的最小整数值 |
| floor(Value) | 小于或者等于 Value 的最大整数值 |
| round(Value) | Value 四舍五入得到的整数值 |

(4) 其他

Math 对象的成员函数还有：max(Value1，Value2)，用于比较两数值，得到 Value1 和 Value2 中较大的值。min(Value1，Value2)用于得到 Value1 和 Value2 中较小的值。

还有一个是用来产生随机数的成员方法。它的语法是：

```
Math.random();
```

它没有任何参数。在 JavaScript 中，每次载入相同的网页，产生的随机数(序列)是不相同的。例如：document.write(Math.random())；将显示不同的随机数。

### 5.5.3　Date 对象

Date(日期)对象封装了有关时间和日期的一些变量和函数，利用这些变量，可以掌握当前日期和时间。Date 对象中没有一个属性是可以直接地设定或者取得的，这种思想更符合对象的封装思想，即成员变量都是私有的，提供公有的成员函数来访问私有的成员变量。

Date 对象的一般格式是：

```
Date(year, month, day, hours, minutes, seconds)
```

#### 1. 创建日期对象

创建日期对象和数组对象有些相似，都需要使用 new 关键字，但它的构造函数比较复杂，有多个不同参数的构造函数，我们可以根据需要选择一种格式来生成一个新的日期对象，下面分别给出它们的语法和范例：

格式 1：var DateName = new Date()；

这是最简单的一种日期对象格式。创建该对象时，激活默认的构造函数，该构造函数取计算机的当前时间作为日期对象的时间值。

格式 2：var DateName= new Date("月 日，年 小时：分：秒")；

在这种格式中，参数是一个字符串，描述了创建的 Date 对象的各个属性，该字符串需要满足以下情况：

(1) 在月、日之间的空格可以省略，但是在年、时间之间的空格不可以省略，否则会使日期无效。月份必须用英语的 January、Februlary、…、December，不能用数字。

(2) 日、年之间的逗号不能省略。

(3) 时间部分的小时、分、秒间的冒号不可省略。

(4) 当"日"这一项超过当月的天数时，将自动进位换算到下一个月。"小时"这一项超过 24 时也将进位换算成日。

例如：var meetDate = new Date("November 24，2003 11:50:00")；

格式 3：var Date1 = new Date(年，月，日)；

这种格式中也有几点要说明：

(1) 参数是数值，不是字符串，年取后两位。

(2) 当"月"超过 12 或"日"超过当月天数时，将自动进位换算到下一年和月。

(3) "月"参数取值是从 0 到 11，即实际月份比参数大一。

(4) 小时、分、秒将被认为是 0。

格式 4：var DateName = new Date(年，月，日，小时，分，秒)；

这种格式与前一种很相近，例如：var Date1=new Date(96，2，23，20，55，00)；

#### 2. get 方法组

我们不能直接读取 Date 对象的属性，但是可以利用一系列 get 方法来取得需要的

信息。

(1) getYear():返回整数表示的日期对象的年份值。如果这个年份是1900年之后的某年,返回的将是后两位,例如1999将返回99;如果是100～1900之间的年份,将返回完全值,例如1769将返回1769。

(2) getMonth():返回整数表示的月份值。取值为0～11,0代表1月,1代表2月等。

(3) getDate():返回整数表示的对象日期是在当月的第几天。1表示是第一天(1号)。

(4) getDay():返回整数表示的对象的日期是星期几。它的数值在0～6,0代表周日,1代表周1,……

(5) getHours():返回整数表示的对象时间的小时值,它的数值是0～23。

(6) getMinites():返回整数表示的对象时间的分钟值,它的数值是0～59。

(7) getSeconds():返回整数表示的对象时间的秒值,它的数值是0～59。

### 3. set方法组

利用set方法,可以设定Date对象的属性,这和get方法组使用类似,并且几乎是一一对应的。

(1) setYear(Year):设定对象的年份值为Year,一般用四位整数。

(2) setMonth(Month):设定对象的月份值为Month,应取值在0～11范围内。

(3) setDate(Day):设定对象在这一月的日期号数为Day,应取值在0～30范围内。

(4) setHours(Hour):设定对象的小时值为Hour,应取值在0～23范围内。

(5) setMinutes(Minute):设定对象的分钟值为Minute,应取值在0～59范围内。

(6) setSeconds(Second):设定对象的秒值为Second,应取值在0～59范围内。

**【例5-4】** 下面是一个利用Date对象显示一个时钟的代码,用来展示Date对象的应用。

```
<html>
<head>
<script language = "javascript">
var timeStr, dateStr;
function timeclock()
{
  now = new Date();
  //时间
  hours = now.getHours();
  minutes = now.getMinutes();
  seconds = now.getSeconds();
  timeStr = "" + hours;
  timeStr + = ((minutes < 10) ? ":0" : ":") + minutes;
  timeStr + = ((seconds < 10) ? ":0" : ":") + seconds;
  document.myclock.time.value = timeStr;
  // 日期
  day = now.getDate();
  month = now.getMonth() + 1;
  month = ((month < 10) ? "0" : "") + month;
  year = now.getYear();
```

```
    dateStr = "" + month;
    dateStr + = ((day < 10) ? "/0" : "/") + day;
    dateStr + = "/" + year;
    document.myclock.date.value = dateStr;
    Timer = setTimeout("clock()",1000);
  }
</script>
</head>
<body onload = "timeclock()">
  <form name = "myclock">
      时间：<input type = "text" name = "time" size = "10" value = ""><br>
      日期：<input type = "text" name = "date" size = "10" value = "">
  </form>
</body>
</html>
```

显示结果如图 5-6 所示。

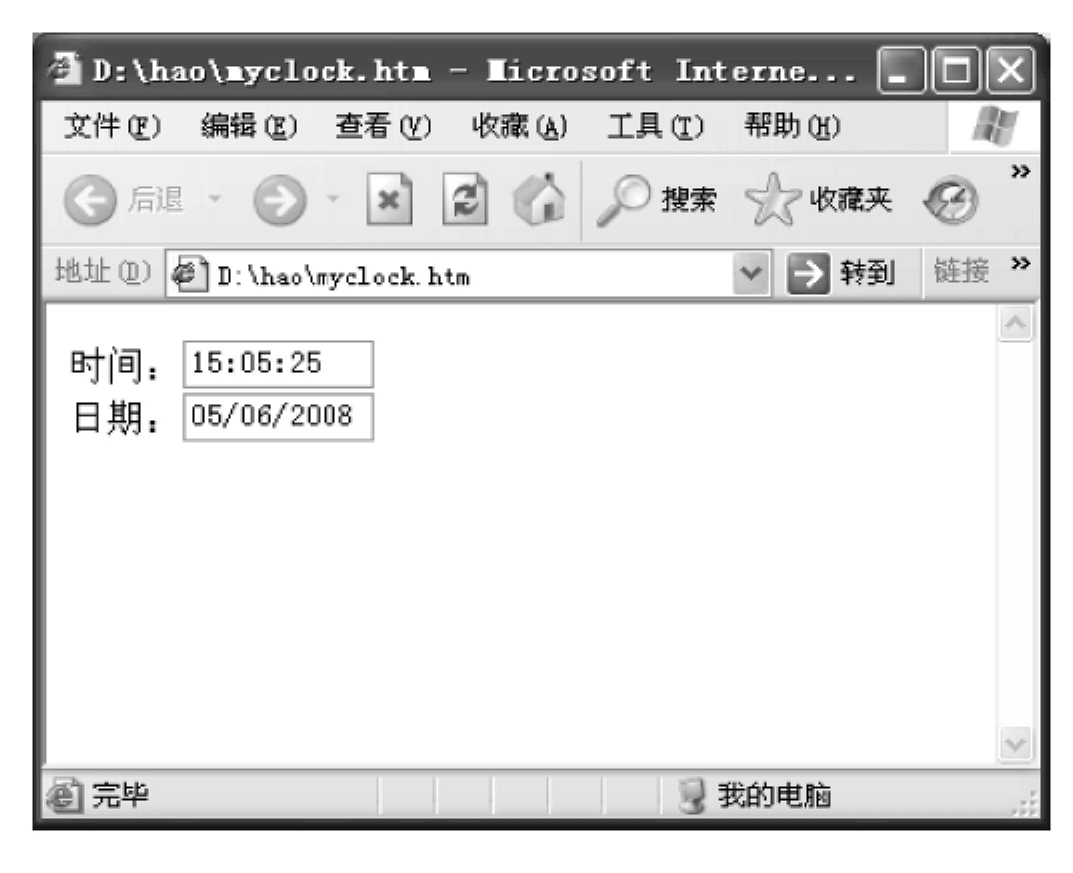



图 5-6 时钟显示示例

## 5.5.4 Array 数组对象

与其他的高级语言不同，JavaScript 没有提供明显的数组类型。数组是一种内置的 Array 对象，它是一组元素对象的集合，元素可以是不同类型的。数组的每一个成员对象都有一个"下标"，用来表示它在数组中的位置(下标从 0 开始)。

### 1. 一维数组

在 JavaScript 中，要使用数组，必须要创建 Array 对象，一般形式是：

```
var <数组名> = new Array();
```

这样就定义了一个空数组，数组中元素的个数不确定，接下来可以为数组添加元素。

一般形式是：

```
<数组名>[<下标>] = <表达式>;
```

例如：

```
var colorArray = new Array();   // 定义一个空数组
colorArray[0] = "red";
colorArray[1] = "green";
colorArray[2] = "blue";
colorArray[3] = 255;
```

另外，用户还可以在定义数组的时候直接初始化元素数据，一般形式是：

```
var <数组名> = new Array(<元素 1>, <元素 2>, <元素 3>, …);
```

例如，var myArray = new Array(1, 4.5, "Hi "); 定义了一个数组 myArray，里边的元素分别是：myArray[0] ==1; myArray[1] ==4.5; myArray[2] =="Hi"。

但是，如果元素列表中只有一个元素，而这个元素又是一个正整数的话，这将定义一个包含正整数个空元素的数组。例如，var myArray = new array(10)，定义一个长度为 10 的数组，而不是一个数组，包含一个正整数元素 10。

### 2. 多维数组

JavaScript 只有一维数组，不能和一般的程序设计语言一样，使用类似 var myArray= new Array(3,4)的方法来定义 3×4 的二维数组，或者用 myArray[1,1]来访问"二维数组"中的元素。实际上，所谓多维数组，就是数组的元素本身也是一个数组。因此，要使用多维数组，在 JavaScript，可用下面的形式来定义：

```
var myArray = new Array(new Array(), new Array(), new Array(), …);
```

例如，要定义一个 3×4 的二维数组，定义如下：

```
var myArray = new Array(new Array(4), new Array(4), new Array(4));
```

然后，可以用 myArray[i][j]的形式访问"二维数组"中的元素。

在 JavaScript 中，数组中的元素可以是不同类型的，因此可以定义 oneArray = new Array(new Array(3), new Array(4)); ，它不是一个 3×4 的二维数组，实际上 oneArray 有两个元素，第一个元素是一个长度为 3 的数组，第二个元素是一个长度为 4 的数组。

### 3. Array 对象的属性

Array 对象常用的属性是 length 属性，保存数组的长度，即数组里元素的个数。它等于数组里最后一个元素的下标加 1。因此，想添加一个元素，可写作：myArray[myArray.length] = …对于二维数组，例如上面的 myArray，document.write(myArray.length)将返回 3，而不是返回 3×4=12。也就是说 myArray 有三个元素，都是长度为 4 的数组。

### 4. Array 对象的方法

(1) join(<分隔符>)方法：把数组中的各个元素串起来，用<分隔符>作为元素之间的分隔符，然后返回这个字符串。该方法不影响数组中的内容。

(2) reverse()方法：使数组中的元素顺序反过来。如果对数组[1，2，3]使用这个方法，它将使数组变成：[3，2，1]。

(3) slice(<s>[，<e>])方法：返回一个从第 s 个元素到第 e 个元素的子数组，如果不给出 e，则返回的子数组从第 s 个元素到数组的最后一个元素。

(4) sort([<方法函数>])方法：使数组中的元素按照一定的顺序排列。如果不指定<方法函数>，则按字母顺序排列。如果指定<方法函数>，则按<方法函数>所指定的排序方法排序。

我们已经学习了四个 JavaScript 中的内置对象，即 String 对象、Math 对象、Date 对象和 Array 组对象。在 JavaScript 的较新版本中，还有一些其他的内置对象，如：函数对象、数值对象、布尔对象等。它们的使用比较简单，属性和方法比较少，请参考其他书籍。

### 5.5.5　预定义函数

除了内置对象外，还有一些功能是经常需要使用的，这些功能被定义为标准的内置函数，不属于任何的对象，因此不用再通过引用对象的方式来使用它们。

(1) eval()函数

语法形式是：

```
var Value = eval(字符串);
```

参数中的字符串是一个合法的表达式转化成的字符串形式，此函数将这个表达式求值，然后返回。例如，可以这样使用这个函数：

```
var tempValue = 10;
var str = "30 + 5 * tempValue";
var myValue = eval(str);
```

这样 myValue 变量中的值是整数 80。这个函数的作用在于，如果表达式是一个字符串，要想得到表达式的值，就必须利用 eval()函数分析那个用来存储这个表达式的字符串。下例中将演示一个最简单的计算器。

**【例 5-5】**　一个简单的表达式示例。

```
<html>
<body>
<form name = "CalForm">
  输入：<input type = "text" name = "Input" size = "20" value = "输入一个表达式">
  <input type = "button" value = " = " width = 8
         onClick = "document.CalForm.Result.value = eval(document.CalForm.Input.value)">
  结果：<input type = "text" name = "Result" size = "8" value = "">
</form>
</body>
</html>
```

上述代码的缺点是对浏览者输入的错误无法做任何的控制，同时也无法判断用户的输入。

(2) parseFloat()函数

将一个合法字符串转换为一个浮点数并被返回。例如：

```
var floatStr = "1.6e30";
var floatValue = parseFloat(floatStr);
```

parseFloat()函数从参数字符串的第一个字符开始处理,如果遇到一个合法的表达方式不可能出现的字符时,就把这个字符前的所有字符作为字符串来转换,其后的字符将被忽略。

(3) parseInt()函数

将一个合法字符串转换为一个整数并被返回,一般形式是:

```
var value = parseInt(字符串,[数制基数]);
```

其中,“数制基数”表示转换的数值进制,为可选项。当选择这个参数时,那么按照选择的数制格式来试图转换这个字符串,其他用法与 parseFloat()函数完全相同。

(4) isNaN()函数

函数 isNaN()用于判断一个字符串是否为数字(Not a Number)。isNaN()函数的一般形式是:

```
Res = isNaN(字符串);
```

如果参数字符串为一个数字,则返回 true,否则返回 false。例如:

```
var res = isNaN("111")
document.write(res);
if (isNaN("hello"))
    {
    window.alert("不是数字字符串");
    }
```

上述代码将输出 false,然后打开一个提示窗口,显示“不是数字字符串”。

## 5.6 JavaScript 浏览器对象模型 BOM

每一个符合 ECMAScript 规范的脚本程序设计语言都包含三个部分,即:ECMAScript 规范、浏览器对象模型(Browser Object Model,BOM)和文档对象模型(Document Object Model,DOM)。所谓浏览器对象模型 BOM,就是指当用户打开浏览器时,浏览器中的 JavaScript runtime engine 将在内存中自动创建一组对象,用于对浏览器及 HTML 文档对象模型中数据的访问和操作。因为这些对象是和浏览器本身紧密相关的,称为浏览器对象。

### 5.6.1 BOM 层次结构

在 JavaScript 中,浏览器对象模型(Browser Object Moel,BOM)定义了浏览器对象的层次结构,如图 5-7 所示。

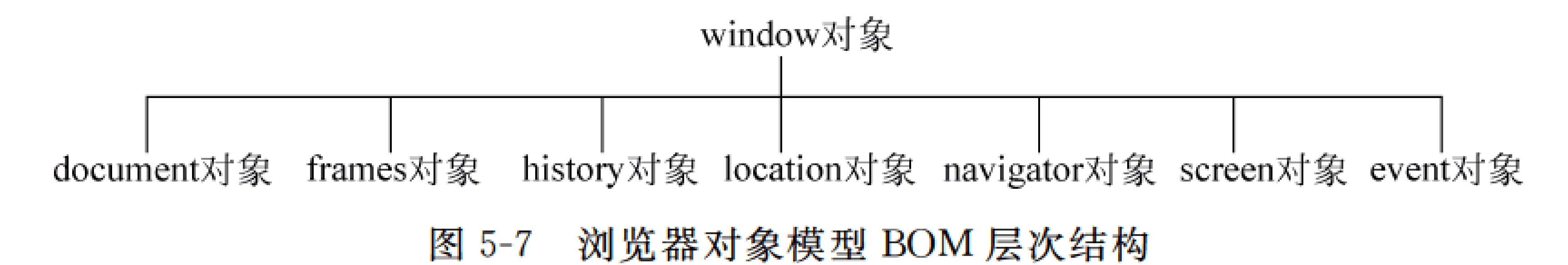



图 5-7 浏览器对象模型 BOM 层次结构

在 JavaScript 中，浏览器对象模型 BOM 的 7 个对象中，window 对象是最顶层的对象，它对应了浏览器窗口本身，其他 6 个对象均是 window 对象的成员对象，其中 document 成员对象也是 HTML 文档对象模型 DOM 中的重要对象。

浏览器对象模型中的每一个对象其实都是一种预声明的内存对象，用户可直接使用。需要注意的是，这些对象的名字的第 1 个字母都是小写的，写成大写会出现 JavaScript 运行错。

## 5.6.2　window 对象

window 即客户的浏览器窗口，window 对象用于描述窗口的特征，它是 document、location、history 等对象的父对象，包含了大量的属性和方法，介绍如下。

### 1. window 对象的常用属性

window 对象的属性可分为一般属性和对象属性，对于每一个属性，具有不同的读写权限，即有的属性为只读属性，不能为属性进行赋值，有的属性则可以进行读写操作。window 对象的一般属性包括：

(1) window、self、parent 和 top

严格地说，window、self、parent 和 top 不能算是 window 对象的属性，更合理的说法是它们是当前环境所涉及的 window 对象实例，或称它们是特殊的关键字。

self 和 window 所代表的都是当前的窗口，这在功能上发生了重叠，一般也不使用。使用 self 和 window 属性的好处是增加程序的可读性，因此在比较复杂的程序中可以考虑使用它。

parent 所指的是当前框架或窗口所在的父窗口，这一属性在使用框架的网页中用途最广泛。top 是用以实现所有下级窗口的窗口，即主窗口，和 parent 相似。

(2) opener 属性

opener 是父窗口的名字。在一个窗口 a 中通过 open()方法打开一个窗口 b，则 b 窗口的 opener 为窗口 a。

(3) screenLeft 和 screenTop 属性

存储窗口的左上角坐标。

(4) defaultStatus 和 status 属性

defaultStatus 是显示在浏览器窗口状态栏上内容的默认值，status 是在浏览器窗口状态栏上内容的当前值。

在刚载入文档时 status 和 defaultStatus 都为空值。在网页中我们所期望的一般是当鼠标在某个超链接或锚点上经过时，显示一条提示信息，这样的用法可以写成：

```
<a href = URL onMouseOver = "status = '欢迎光临'; return true">
```

一般情况下，status 属性与 onMouseOver 同时使用。改变 status 属性直接影响状态栏的显示，如果与 onMouseOver 同时使用，应使此事件处理程序返回 true。当鼠标位于文档窗口内，并且不在任何的超链接和锚点上时，显示 defaultStatus 属性的值。

除了上述的一般属性外，window 对象还包含一组成员对象，包括：navigator 对象、document 对象、frames 对象、history 对象、location 对象、screen 对象等。

### 2. window对象的常用方法

(1) 打开和关闭窗口

① open()方法：打开一个新窗口，一般形式是：

```
open("URL","WindowName","Window Features");
```

其中，参数URL表示打开新窗口中要打开的网页，参数WindowName表示窗口的名字，第三个参数是可选的，它表示新窗口的外观，是一个字符串，可以指定多个特性，每一个特性间用逗号分隔。

open()方法返回一个指向该窗口对象的指针，通过该指针可以访问新打开的窗口。

② close()方法：用来关闭一个窗口。如果关闭当前窗口，直接使用这个方法即可；如果要关闭一个由JavaScript打开的其他窗口，则需要在打开该窗口时保留对这个窗口的引用，然后使其调用自身的close()，如：newWin.close()。

(2) 移动窗口位置

在JavaScript中，定义屏幕左上角(x,y)坐标为(0,0)，水平方向为X轴的正方向，垂直方向为Y轴正方向。

moveBy(x,y)方法将浏览器窗口分别在X轴和Y轴方向上移动x,y个像素单位。moveTo(x,y)方法则将浏览器窗口左上角移动到(x,y)坐标处。

(3) 改变窗口大小

resizeBy(x,y)方法将窗口在x,y方向进行放大和缩小x,y个像素值，窗口的左上角坐标不变。resizeTo(x,y)方法将窗口的宽度和高度调整为x,y，窗口左上角位置不变。

(4) 窗口滚动

scroll(x,y)方法有两个参数x,y，它使窗口卷滚到x,y坐标处。

(5) focus()和blur()方法

focus()方法使某个窗口获得焦点，获得焦点的体现就是它出现在最前面。blur()方法使某个窗口失去焦点，体现在它被发配到后台。

(6) 系统提示相关方法

① alert(字符串)方法：显示一个窗口。

② confirm(字符串)方法：是一个确认形式的对话框，但它是有返回值的。这个方法调用后显示一个有“确定”和“取消”按钮的对话框，函数返回值是一个布尔值。如果单击“确认”按钮，则返回true；单击“取消”按钮，返回false。

③ prompt()方法：上面的两个方法都无法从用户那里得到比Yes和No更多的信息。而prompt()包含两个参数，其语法为：

```
prompt("message","initial input");
```

其中message参数的作用类似于alert()的参数；initial input是输入空白处的默认值；函数调用的返回值是一个字符串。可以这样调用：

```
msg = prompt("请输入您的电话号码?","1234567");
```

对话框内输入电话号码后单击“确定”按钮，那么msg这个变量内就保存了你的电话号

码；相反，如果单击“取消”按钮，msg是null。

(7) setTimeout()和clearTimeout()方法

对于Date对象，可以显示当前时间，而无法根据时间改变动态地进行显示。JavaScript还提供了另外两个函数：setTimeout()和clearTimeout()。

setTimeout()译作“超时”，它的语法是：

```
timeID = setTimeout("expession",time);
```

setTimeout()运行后，在time时间过后，将对“expression”求值一次，但并不是每过一个time时间就求值一次，因此要想周期地改变，则需要周期地使用这一方法。在实际使用中，“expression”参数通常是一个函数，setTimeout()的返回值是一个整数，用以标记一个特定的“超时”。clearTimeout()的参数只有一个，就是上面的timeID，通过传递此参数才可以确定要清除哪一个超时设置。

**【例5-6】** window对象应用举例。

为了学习window对象的属性和方法，在本例中定义四个按钮：“打开新窗口”、“关闭新窗口”、“查看源码”和“测试window对象属性及方法”。单击“打开新窗口”按钮，则打开一个新窗口，单击“查看源码”窗口则显示当前Web页的HTML源代码。假设窗口的特征是没有工具栏、菜单、状态栏、定位栏，高200个像素，宽200个像素。按钮“测试window对象属性及方法”对应一个test函数，通过修改该函数内容，可以分别测试window对象的其他属性和方法。测试页面文件名testwin.htm，HTML代码如下：

```
<html>
<head>
<script language = "javascript">
function newone(url)
{
  newWin = open(url,"subWindow", "width = 200,height = 200,toolbar = 0,status = 0,location = 0,
menubar = 0");
  return newWin;
}
function closeone(xWin)
{
  // 若窗口存在,并且没有被关闭
  if (xWin ! = "" && ! xWin.closed)
        xWin.close();
}
// 可以修改该函数代码,测试 window 对象的其他方法
function test()
{
  window.moveTo(0,0);
}
</script>
</head>
<body>
  <input type = "button" value = "打开新窗口" name = "B1" onclick = "javascript:temp = newone
('bookIndex.htm')" > 
  <input type = "button" value = "关闭新窗口" name = "B2" onclick = "javascript:closeone
```

```
(temp);" ><br>
  <input type = "button" value = "显示源文件" name = "B3"
         onclick = "window.location = 'view - source:' + window.location.href"><br>
  <input type = "button" value = "测试 window 对象属性及方法" name = "B4" onclick = "test()" >
</body>
</html>
```

新窗口对应的 HTML 文件 bookIndex.htm 内容如下：

```
<html>
<head>
<meta http - equiv = "Content - Type" content = "text/html; charset = gb2312">
<title>book Index</title>
</head>
<body>
<p align = center>Index for The book</p>
<hr>
<input type = "button" value = "关闭窗口" name = "B1" onclick = "window.close()" >
</body>
</html>
```

在浏览器中打开 testwin.htm 文件，单击“打开新窗口”按钮，结果如图 5-8 所示。

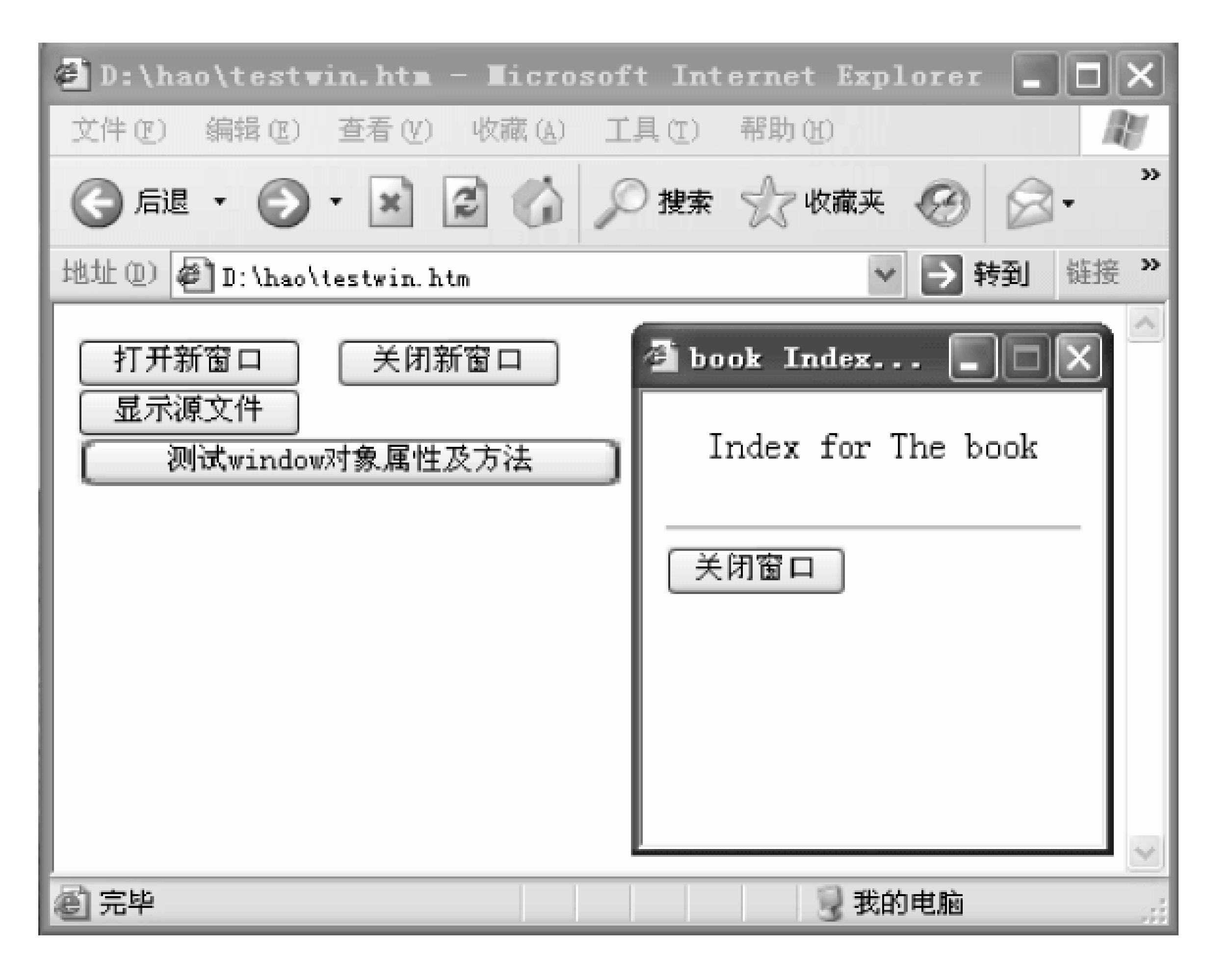



图 5-8 window 对象测试示例

**【例 5-7】** 新建一个窗口，不显示任何 HTML 文档，改变窗口焦点状态示例。

```
<html>
<title>The operation of window</title>
<body>
<form>
   <input type = "button" value = "新建窗口"
         onClick = 'newWin = open("","subWindow","height = 200,width = 200")'><br>
   <input type = "button" value = "窗口前台" onClick = "newWin.focus()"><br>
   <input type = "button" value = "隐藏窗口" onClick = "newWin.blur()">
</form>
```

```
</body>
<html>
```

上述代码可以完成新建窗口，以及实现新窗口前台和后台显示的转换。

**【例 5-8】** 网页中时钟的实现示例。

```
<html>
<head>
<script language = "javascript">
   var timeID;
   var Running = false;
   function StartTime()
   {
     running = true;
     ChangeTime();
   }
   function ChangeTime()
   {
     var currentTime = new Date();
     document.myClock.show.value = currentTime.toLocaleString();
     //递归调用
     timeID = setTimeout("ChangeTime()",1000);
    }
   function StopTime()
   {
      if (Running)
         clearTimeout(timeID);
      Running = false;
   }
</script>
</head>
<body onLoad = "StartTime()" onUnload = "StopTime()">
<form name = "myClock">
    <input type = "text" name = "show" value = " " size = 20>
</form>
</body>
</html>
```

上述代码在浏览器中显示结果为：

2004-07-31 13:06:19

上面介绍了 window 对象的常用属性和方法，在 FrontPage 中，在“代码”视图下，在输入“window.”后自动打开 IntelliSense 窗口，可以显示所有的 window 对象属性和方法。需要注意的是，window 对象是任何对象、属性和方法的假定父对象。也就是说，用户可以在任一对象、属性和方法(当然不包括 window 对象本身)之前加上一个“window.”，作为它的所有者。如果不加的话，浏览器仍假定其所有者为 window 对象。

例如，语句 alert("hello，world")；和 window.alert("hello，world)；是等同的。

### 5.6.3 navigator 对象

在 BOM 中，navigator 对象是 window 对象的成员对象，它封装了有关操作系统、浏览器版本等环境信息，主要属性和方法有：

(1) appName 属性：存储表示浏览器名称的字符串。如果使用的是 IE 或遨游浏览器，appName 中的值为"Microsoft Internet Explorer"。如果客户使用的是 Netscape 或 Mozilla Firefox，则这个字符串的值是"Netscape"。

(2) appVersion 属性：存储客户所用浏览器的版本号和操作系统等信息，不同的浏览器显示的内容项目不同。

(3) appCodeName 属性：浏览器代码名称，只读属性。不论是 IE、Maxhon，还是 Mozilla Firefox 浏览器，appCodeName 具有相同的值，都是 Mozilla。

(4) javaEnabled()方法：这个函数返回一个布尔值 true 或 false，它表示 Java 是否能在这个客户端浏览器内使用。

(5) plugins 属性：在 Navigator 中可以使用的 plugins、mimeType 和 plugins 存放在两个数组中，这两个数组分别由 mimeTypes 和 plugins 对象组成。

(6) cookieEnabled：说明浏览器是否打开了 cookies，只读属性。

(7) userAgent 属性：浏览器的完整的用户代理标志，这是 appName、appVersion 和 appCodeName 组合的结果。例如"Mozilla/3.04Gold(Win95；I)"。

(8) platform：给出运行浏览器的操作系统或硬件的类型。

(9) systemLanguage：给出操作系统使用的默认语言，只读。

**【例 5-9】** 使用 navigator 对象查看客户端浏览器特性。

```
<html>
<head>
<script language = "JavaScript">
function TrueFalse (flag)
{
   return (flag? "是":"否");
}
</script>
</head>
<body>
<h2 align = center>您的浏览器特性</h2>
<script language = "JavaScript">
   document.write("        浏览器名称：" + navigator.appName + "<br>");
   document.write("        浏览器版本：" + navigator.appVersion + "<br>");
   document.write("    浏览器代码名称：" + navigator.appCodeName + "<br>");
   document.write("      支持 JavaScript：" + TrueFalse(navigator.javaEnabled()) + "<br>");
</script>
</body>
</html>
```

上述代码在 Windows XP 操作系统中的 IE 6.0 浏览器中的显示结果如图 5-9 所示。




图 5-9　浏览器特性

每个浏览器对象都包含一组属性和方法，在 FrontPage 的代码视图中，输入对象名和后面的字符“.”后，将打开 IntelliSense 智能帮助窗口，显示当前对象的成员变量和成员函数。

## 5.6.4　frames 对象

在 window 属性中，frames 是一个 array 型成员对象，如果窗口分帧的话，每个 frame 相当于一个单独的 window 对象。对帧的访问可以通过数组下标 0，1，…，或者通过帧名。例如，window.frames[0]代表第一个帧窗口。

**【例 5-10】**　帧的操作。建立一个包含左右两个帧的页面，在左面帧页中，设置一个 button 按钮，单击该按钮，改变右侧帧窗口的背景色。

帧代码清单(myfrme.htm)如下：

```
<html>
<head>
<meta HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=gb2312">
<title>新建网页 1</title>
</head>
<frameset cols="30%,*">
<frame name="left" src="myleft.htm">
<frame name="right" src="myright.htm">
<noframes>
<body>
<p>此网页使用了框架，但您的浏览器不支持框架。</p>
</body>
</noframes>
</frameset>
```

左侧帧窗口(left)代码清单(myleft.htm)如下：

```
</html>
```

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>left</title>
<script>
// 将帧窗口 fname 的背景色设置为 color
// fname 的取值可以是数字 0,1,…,也可以为帧的名称
function test(fname,color)
{
 window.parent.frames[fname].document.bgColor = color;
}
</script>
</head>
<body>
<p align="center">
<input type="button" value="改变右边帧背景色" name="B4" onclick=test("right","red")>
</p>
</body>
</html>
```

### 5.6.5 location 对象

location 对象用来存储当前窗口的 URL 值。它存储了一个完整的 URL,并且可以通过对 location 对象的赋值来改变当前窗口的 URL,从而改变当前网页的内容。

location 对象包含的属性和方法如表 5-6 所示。

**表 5-6 location 对象常用属性列表**

| 属性或方法 | 说　明 |
|---|---|
| host 属性 | 存储 URL 的主机名和端口号,只有端口号是 URL 的一个明确部分时,值中才包括端口号 |
| hostname 属性 | 存储 URL 地址中的主机名+域名部分,不包括端口号 |
| href 属性 | 存储窗口对象的整个 URL 的字符串 |
| pathname 属性 | 存储 URL 中文件路径。如果 URL 中的网页文件是根下的一个文件,则 location.pathname 的值为根(/) |
| port 属性 | 存储 URL 中的端口号 |
| protocol 属性 | 存储 URL 中的协议名,包括后面的冒号(:) |
| assign 方法 | assign("URL")通过这个方法可以实现把一个新的 URL 赋给 location 对象。也可以通过为 location.href 赋值来导航到一个新的网页。采用 assign 的方法会使代码易维护 |
| reload 方法 | 重新装载当前文档 |
| replace("URL")方法 | 装载一个新文档但不创建一个新的历史记录,即:在浏览器的历史列表中,新文档将替换当前文档 |

我们在介绍 document 对象时曾提及过一个 location 属性,我们要将这两个同名者区分开。只需要记住 location 对象是可读可写的,就是说,我们可以对 location 对象的属性赋

值；而 document 对象的 location 属性是一个只读属性，它只是提供文档的 URL，改变它也是没有意义的。因为 document 的 location 属性描述了文档的 URL，改变它就是表示这个文档被复制到另外的 URL，这是不可能发生的。

### 5.6.6 history 对象

一般把 history 对象称作历史清单对象，它保存窗口或框架在某时间段内的 URL。出于安全方面的原因，history 对象并不向脚本提供列表的真正内容，例如一个完整的 URL 名称。对这样的 URL 是无法得到这个具体的字符串的，也就是说无法通过它来得到用户曾经访问的 URL 的具体值。这是为了防止恶意的脚本程序获得用户个人的浏览习惯而进行不正当的网络传输或散播，这些信息是大部分用户不愿意泄露的。

history 对象的属性和方法较少，且相对简单，history 对象属性和方法见表 5-7。

**表 5-7　history 对象的属性和方法**

| 方　法 | 描　述 |
|---|---|
| length | 反映历史清单的长度，为只读属性 |
| back() | 装载历史清单中的前一个 URL，没有参数 |
| forward() | 装载历史清单中的后一个 URL，没有参数 |
| go() | 参数既可以是整数，也可以是字符串。当参数是整数 x 时，装载历史清单中当前位置偏移 x 后的 URL。当参数是字符串时，装载历史清单中含有这个字符串的最近 URL |

如果我们的网页有很多内容，那么靠浏览器提供的 Back 按钮返回前一个 URL 较慢，可以在网页中设置"快速返回"按钮，代码如下：

```
<input type='button' value="快速返回" onClick="history.go(-5)">
```

在每页中的适当位置设置这个按钮，每按一次将向后 5 个 URL，结合浏览器的 Back 按钮，将可以快速地向后翻动。

### 5.6.7 screen 对象

window 对象的 screen 成员对象包含有关客户机显示屏幕的信息。JavaScript 程序可以利用这些信息来优化输出，以达到用户的显示要求。例如，一个程序可以根据显示器的尺寸选择使用大图像还是使用小图像，或者根据屏幕尺寸将浏览器窗口定位在屏幕中间。

在 screen 对象中，常用的属性是：availHeight(显示屏幕的高度)，availWidth(显示屏幕的宽度)，height(显示器屏幕的高度)，width(显示器屏幕的宽度)。

**【例 5-11】** 利用 screen 对象，在一个窗口中打开一个窗口，让窗口在屏幕上平滑地移动，遇到屏幕边界，将自动弹回，改变移动方向。代码清单(bouncewin.htm)如下：

```
<html>
<head>
<script>
// 初始化动画的变量
```

```
var x = 0, y = 0, w = 100, h = 100;       // 窗口的位置和大小
var dx = 5, dy = 5;                        // 窗口在 x、y 方向每次移动的距离
//窗口移动函数,将窗口移动到(x + dx, y + dy)位置
function bounce()
{
     if (win.closed)
     {
          window.clearInterval(winID);
          return;
     }
     if ((x + dx > (screen.availWidth - w)) || (x + dx < 0))
           dx = - dx;
     if ((y + dy > (screen.availHeight - h)) || (y + dy < 0))
           dy = - dy;
     //更新窗口的位置
     x + = dx;
     y + = dy;
     win.moveTo(x,y);
}
</script>
</head>
<body>
<script>
//打开一个子窗口
var win = window.open('javascript:"<h1>hello</h1>"',"bounce","width = " + w + ",height = " + h);
// 窗口的初始位置
win.moveTo(x,y);
// 使用 setInterval() 每隔 100 毫秒调用 bounce()
var winID = window.setInterval("bounce()", 100);
</script>
<form>
<input type = "button" value = "Stop"
          onclick = "clearInterval(winID); win.close();">
</form>
</body>
</html>
```

在上述页面中,打开一个窗口,并启动 window 对象的时间间隔函数,开始移动窗口。在父窗口中,单击按钮 stop,移动的窗口将被关闭。

### 5.6.8 event 对象

要提高网页的交互性,必须要了解 event 对象。它在事件创立时产生,如:单击一个可点击的对象,移动鼠标,或获得输入焦点等,此时,Event 对象被创建,用于存储按键值、当前的(x,y)坐标等。

### 1. event 对象的属性

在不同的浏览器中，event 对象的属性不完全相同，常见的 event 对象属性见表 5-8。

表 5-8　event 对象属性

| 属 性 名 | 属 性 描 述 | 属 性 名 | 属 性 描 述 |
|---|---|---|---|
| type | 事件类型的字符串 | clientX，clientY | 相对于页面的横坐标和纵坐标 |
| srcElement | 发送给事件对象的字符串 | screenX，screenY | 相对于屏幕的横坐标和纵坐标 |
| x、y | 光标行、列坐标 | keyCode | 键代码 |

### 2. 可能的事件

在浏览器中可能的事件主要有：

onDblClick：鼠标双击。

onKeyDown：键被按下。

onKeyPress：键被按下然后被释放。

onKeyUp：键被释放。

onMouseDown：鼠标被按下。

onMouseMove：鼠标移动。

onMouseUp：鼠标被释放。

onResize：窗口被调整大小。

### 3. 事件处理方法

事件是由事件处理函数进行处理的。但是，对于两种不同的浏览器，事件的处理途径不完全一样。在 IE 中，采用“事件气泡”的方式处理事件，例如：

```
<body onclick = "f1()">
    <p onclick = "f2()">
        <em onclick = "f3()">
            <strong onclick = "f4()">Click on me</strong>
        </em>
    </p>
</body>
```

对于上述的 HTML 代码，每个标记都接受鼠标单击事件，都创建 event 对象。那么，这个事件如何处理呢？顺序是这样的，例如，单击 strong 标记内的文本，它接收到一个 onClick 事件，执行 f4()；然后发送 onclick 事件给<em>标记，执行 f3()；然后发送到<p>标记，执行 f2()；然后直到窗口。这样每个元素分别以自己的方式处理鼠标单击事件。

但是如果想停止事件上传，可以在事件处理函数中添加取消气泡的代码，方法是：

```
function fx ()
{
…;
this.event.cancelBubble = true;
}
```

# 5.7 HTML 文档对象模型 DOM

当浏览器打开一个网页时,不管是 HTML 还是 XML 文档,对于网页中的每一个标记,都在内存中创建一个相应的内存对象。这些对象按照树形结构组织,这就是文档对象模型 DOM。我们可以将 DOM 理解为网页的 API,它将网页中的元素看作一个个对象,这些对象通过标记的 name 属性来命名,通过对这些可访问的内存对象进行编程,来实现对网页中元素及其属性的修改,以便动态地修改网页。

## 5.7.1 文档对象模型 DOM

根据 W3C DOM 规范,文档对象模型(Document Object Model,DOM)是一种与浏览器、平台、语言无关的接口。DOM 解决了 Netscape 的 JavaScript 和 Microsoft 的 JScript 之间的冲突,给予 Web 设计师和开发者一个标准的方法,使程序和脚本能动态地访问和更新文档的内容。

W3C DOM 被分为 3 个不同的部分,即:核心 DOM、XML DOM 和 HTML DOM。核心 DOM 是用于任何结构化文档的标准模型,XML DOM 和 HTML DOM 分别是用于 XML 文档和 HTML 文档的标准模型。在 DOM 中,定义了所有文档元素的对象和属性,以及访问它们的方法(接口)。

对于 DOM 对象的访问,可以出现在页面的脚本程序中,也可以直接在浏览器地址栏中书写。例如,当浏览网页时,有时候需要知道网页的发布时间,从而判断网页内容是否是很久以前的,此时,可以在浏览器的地址栏中输入:

```
javascript:document.write(document.lastModified)
```

输入结束后,按回车键,则打开一个新的网页显示当前正在浏览的网页的最后修改日期。然后单击浏览器工具栏中的后退按钮,返回到刚才浏览的网页。

## 5.7.2 HTML DOM 对象

在 HTML 文档中,当浏览器打开一个网页时,为每一个标记在内存中建立一个内存对象,即 HTML DOM 对象,因此,根据 HTML 规范,常用的 HTML DOM 对象如表 5-9 所示。

**表 5-9 常用的 HTML DOM 对象**

| DOM 对象 | 说明 | DOM 对象 | 说明 |
|---|---|---|---|
| document | 表示整个 HTML 文档,可用来访问页面中的所有元素 | form | 对应<form>元素 |
| meta | 对应一个<meta>元素 | button | 对应<button>元素 |
| link | 对应一个<link>元素 | input radio | 对应表单中的一个单选按钮 |

续表

| DOM 对象 | 说　　明 | DOM 对象 | 说　　明 |
|---|---|---|---|
| style | 对应一个单独的样式声明 | input text | 对应表单中的一个文本框 |
| base | 对应<base>元素 | input password | 对应表单中的一个密码域 |
| body | 对应<body>元素 | textarea | 对应<textarea>元素 |
| image | 对应<img>元素 | input checkbox | 对应表单中的一个复选框 |
| anchor | 对应<a>元素 | select | 对应表单中的一个选择列表 |
| table | 对应<table>元素 | option | 对应<option>元素 |
| TableRow | 对应<tr>元素 | input file | 对应表单中的一个文件上传 |
| TableData | 对应<td>元素 | input hidden | 对应表单中的一个隐藏域 |
| area | 对应图像地图中的<area> 元素 | input submit | 对应表单中的一个确认按钮 |
| frameset | 对应<frameset>元素 | input reset | 对应表单中的一个重置按钮 |
| frame | 对应<frame>元素 | object | 对应<object>元素 |
| iframe | 对应<iframe>元素 | event | 代表某个事件的状态 |

在所有的 HTML DOM 对象中，它们之间为层次结构关系，这种关系和 HTML 规范中标记之间的层次关系一致。因此，可以说 document 对象是所有其他 DOM 对象的父对象，其他对象都是通过 document 对象访问的。HTML DOM 对象层次结构关系如图 5-10 所示。

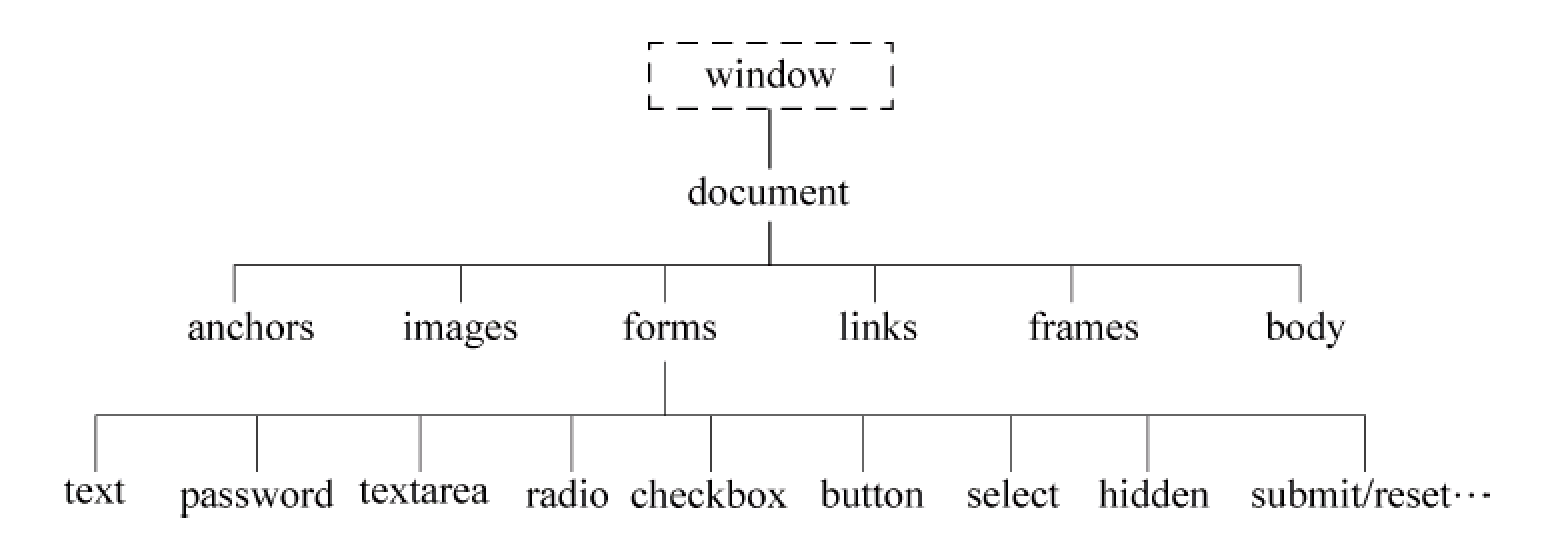



图 5-10　HTML DOM 对象层次结构

下面介绍部分 DOM 对象的属性和方法。

#### 1. document 对象

在 HTML DOM 中，document 对象代表整个 HTML 文档，可用来访问页面中的所有元素，包括文档头<head>部分和文档体<body>部分。此外，需要注意的是 document 对象是浏览器对象模型 BOM 中的 window 对象的一个部分，可通过 window. document 属性来访问。

(1) document 对象的属性

document 对象包含的属性很多，在 FrontPage 中的代码视图中，输入 window. document 可打开 IntelliSense 窗口，显示其所包含的属性和方法。document 对象包括一组集合属性，属性名及含义见表 5-10。

表 5-10　document 对象常用属性列表

| 属 性 名 | 说　　明 | 属 性 名 | 说　　明 |
|---|---|---|---|
| all[]集合 | 提供对文档中所有 HTML 元素的访问 | URL | 返回当前文档的 URL |
| anchors[]集合 | 返回对文档中所有 Anchor 对象的引用 | body | 提供对<body>元素及其属性的直接访问 |
| images[]集合 | 返回对文档中所有 Image 对象的引用 | domain | 返回当前文档的域名 |
| links[]集合 | 返回对文档中所有 Area 和 Link 对象的引用 | title | 返回当前文档的标题 |
| forms[]集合 | 返回对文档中所有 Form 对象的引用 | referrer | 返回载入当前文档的 URL |
| applets | 返回对文档中所有 Applet 对象的引用 | lastModified | 返回文档被最后修改的日期和时间 |

(2) document 对象的方法

document 对象的方法如表 5-11 所示。

表 5-11　document 对象常用方法列表

| 方　　法 | 描　　述 |
|---|---|
| open() | 打开一个输出流，接受来自 document. write()或 document. writeln()方法的输出 |
| close() | 关闭用 document. open()方法打开的输出流，并显示选定的数据 |
| write() | 向文档写 HTML 表达式或 JavaScript 代码 |
| writeln() | 等同于 write()方法，输出结束后输出一个换行符 |
| getElementById() | 返回对拥有指定 id 的第一个对象的引用 |
| getElementsByName() | 返回带有指定名称的对象集合 |
| getElementsByTagName() | 返回带有指定标签名的对象集合 |

### 2. body 对象

每一个 HTML 文档有一个<body>标记，浏览器在内存中创建一个 body 对象，body 对象封装了 HTML 文档中<body>标记的所有属性及所有的文档体元素。从 HTML 结构看，body 对象是 document 对象的成员对象，body 对象的常用属性和方法见表 5-12。

表 5-12　document. body 对象属性列表

| 属　　性 | 描　　述 | 属　　性 | 描　　述 |
|---|---|---|---|
| id | 设置或返回<body>元素的 id 属性 | innerHTML | innerHTML 可以赋值一段符合 HTML 规范的文本，该文本将重写现有网页的<body>体内的内容 |
| title | 设置或返回<body>元素的 title 属性 | innerText | 为 innerText 赋值，页面将用 innerText 的内容重写<body>体内的内容 |

续表

| 属　性 | 描　述 | 属　性 | 描　述 |
|---|---|---|---|
| bgColor | 设置或返回<body>元素的背景色属性 | clientLeft，clientTop | 存储浏览器窗口客户区左上角(x，y)坐标 |
| disabled | 如果该成员变量设为 true，则页面的内容被灰化，且不可被选取，右击时快捷菜单被禁用 | clientWidth，clientHeight | 存储浏览器窗口客户区宽度和高度 |
| alink | 对应<body>标记的 alink 属性，设活动链接的颜色 | topMargin，bottomMargin | 存储浏览器窗口客户区内容和上下边框的距离(像素) |
| vlink | 对应<body>标记的 vlink 属性，设访问过链接的颜色 | leftMargin，rightMargin | 存储浏览器窗口客户区内容和左右边框的距离(像素) |

对于 body 对象的 innerHTML 和 innerText 的属性功能不同，如果一段字符串文本包含 HTML 标记，将该文本赋给 innerText ，则页面内容用 innerText 的文本替换，该文本不按照其包含的 THML 标记显示；如果同样的文本赋给 innerHTML 属性，则页面内容被 innerHTML 中的文本替换，该文本按照其包含的 HTML 标记显示。例如，document. body. innerText＝"<b> Hello </b>"和 document. body. innerHTML＝"<b> Hello </b>"，产生的结果不同。

从 HTML 文档结构看，body 对象是 document 对象的成员对象，对文档体内所有元素的访问应该通过 document. body 来访问。但是，为了书写上的方便，在 document 对象中，直接包含了有关文档体内的元素对象。例如，设置文档的背景色，理论上应该写为：document. body. bgColor＝""，但是，也可以直接写为：document. bgColor＝""，两种写法结果是一样的。实际上，对文档体中大多数元素的访问几乎都可以直接使用 document，而不是使用 document. body。

例如，下面的例子将展示设置 <body> 元素的 id 的两种方法：

```
<html>
<body id = "myid" title = "hello">
<script type = "text/javascript">
   x = document.getElementsByTagName('body')[0];
   document.write("Body id: " + x.id);
   document.write("<br/>");
   document.write("An alternate way: ");
   document.write(document.getElementById('myid').id);
   document.write("Body title: " + x.title);
</script>
</body>
</html>
```

输出结果为：

```
Body id: myid
An alternate way: myid
Body title: hello
```

【例 5-12】 document 对象应用举例。

在 HTML 中,定义书签(又称锚点)是用<a name="">标识来定义的,按照它们在 HTML 文档中的定义顺序,document 对象的 anchors 数组成员记录了这些锚点。anchors 数组最简单的用法是采用框架分别开设两个窗口,一个是浏览导航窗口,另一个是主浏览窗口。在浏览导航窗口中根据锚点的数量设置按钮,单击某个按钮即可在浏览窗口中跳到特定的文字处。

下面通过一个包含两个帧的框架来说明 document 对象中 anchors 的应用,对应的三个 HTML 文档分别为:main. htm、mainA. htm 和 mainB. htm。

(1) 框架页面文档 main. htm

```
<html>
<frameset cols = " * , * ">
      <frame src = "mainA. htm" name = "FrameA">
      <frame src = "mainB. htm" name = "FrameB">
</frameset>
</html>
```

上述 HTML 文档将浏览器窗口分成两部分,左边一个用作导航,右边一个用作浏览。导航窗口包括几个按钮,它们表示文章中几个可以利用的锚点。

(2) 在导航窗口,首先计算浏览窗口中锚点的个数,然后再设定导航按钮的个数。

mainA. htm 文档:

```
<html>
<script language = "javascript">
function SearchAnchor(Index)
{
  parent.FrameB.location.hash = "part" + Index;
}
</script>
<body>
<script language = "javascript">
var Num = parent.FrameB.document.anchors.length;
for (var i = 1;i< = Num;i++)
{
    document.write("<input type = \'button\' value = \'第",i,"部分")
    document.write("\' onClick = \'SearchAnchor(");
    document.write(i);
    document.write(")\'>");
    document.write("<br>");
}
</script>
</body>
</html>
```

上述代码中 parent. FrameB. location. hash 是 location 对象的内容,在后面介绍。

(3) mainB. htm 文件不包含任何 JavaScript 内容,它定义了在 FrameA. htm 中引用的必要的锚点,注意锚点的名字是 part1～part3。FrameB. htm 的源文件如下:

mainB. htm 文档:

```
<html>
<body>
<a name = "part1"><b>第一部分 四大名著</b></a>
<ul>
<li>红楼梦
<li>水浒传
<li>西游记
<li>三国演义
</ul>
<a name = "part2"><b>第二部分 金庸小说</b></a>
<ul>
<li>笑傲江湖
<li>射雕英雄传
<li>天龙八部
</ul>
</body>
</html>
```

### 3. images 数组成员属性

images 数组的每个元素都是 image 对象。首先看 HTML 中与<img>标签相关的属性,例如<img name="plane" src="plane.gif" hspace="20" vspace="20" width="30" height="50" border="5" lowsrc="plane1.gif">。

这个 HTML 语句中用了一些平时不会经常用到的属性设置,image 对象的成员属性相关描述见表 5-13。

**表 5-13 image 对象的成员属性**

| 属　性　名 | 描　　述 |
|---|---|
| name | image 对象的名字,供脚本程序访问 image 对象使用 |
| src | image 对象对应图片的 URL |
| lowsrc | 存储真正的图片还未装载之前,可以先装载的一幅图片的 URL,该图片一般是实际图像的低分辨率版本 |
| border | 图片周围边框的宽度,默认值为 0 |
| height, width | 图片的高度和宽度 |
| vspace | 图片在垂直方向上与上面或下面文字之间的距离 |

在网页中使用图片,除了增加页面的视觉效果,还可以通过 JavaScript 程序动态地选择载入的图片,或者实现一些动画效果,从而增加页面的动感。

**【例 5-13】** 动画显示。

```
<html>
<head>
<script language = "javascript">
var ImageNum = 1;
function Begin()
{
  document.MyImage.src = ImageArray[ImageNum].src;
```

```
  ImageNum++;
  If (ImageNum>3)
     ImageNum = 1;
}
</script>
</head>
<body>
<img name = "MyImage" src = "tu.gif" onLoad = "setTimeout('Begin()',500)">
<script language = "javascript">
var ImageArray = new Array();
for (i = 1;i< = 3;i++)
{
  ImageArray[i] = new image();
  ImageArray[i].src = "tu" + i + ".gif"
}
</script>
</body>
<html>
```

onLoad 事件设置了 500 毫秒之后运行 Begin()函数,装入下一幅图片后又引起 onLoad 事件,因此 500 毫秒后又运行 Begin()函数,如此循环就实现了一个最简单的动画。

另外,和 Image 对象有关的事件包括:

- onAbort 事件,当用户放弃载入一个图像时触发一个 abort 事件,于是会执行 onAbort 事件处理函数的 JavaScript 代码。例如,当在载入的过程中单击一个链接或是单击了浏览器中的 Stop 按钮时,就是这种情况。
- OnError 事件,当浏览器完成载入一幅图像时触发一个 load 事件,注意,是完成载入后,而不是载入开始时。

#### 4. links 链接数组

链接数组是另一种常用的 document 对象中的数组成员,links 数组的每一个元素都是一个 link 对象,或者是 area 对象,对应 HTML 文档中的一个超链接标记<a>,用来存储 URL 的信息,通过 links 数组可以对文档中的超链接进行操作。一个完整的 URL 可以分成几个组成部分,例如下面的 URL:

```
http://www.abc.xyz.cn:8080/javascript/index.html#start
```

http 是协议部分,表示使用超级文本传输协议 http,www 是主机名,abc. xyz 是主机所在的域名,这一部分也可能是数字表示的 IP 地址;8080 是主机用于当前连接的端口号;/javascript/index. html 是文档路径;#start 对应文档中的书签,用于定位文档中的 anchor。

对应 URL 的各个部分,link 对象中有若干个属性与此对应,分别是:

host 定义网络主机名、域名或 IP 地址,在上面的例子中是 www. abc. xyz. cn。hostname 是主机和端口的组合,在上面的例子中是 www. abc. xyz. cn:8080。href 代表整个 URL。pathname 是路径,在上面的例子中是/javascript/index. html。port 是服务器端口号,在上面的例子中是 2001。protocol 是协议部分,在上面的例子中是"http"。一个链接包括 URL 的所有信息。上面的这些 link 对象成员属性将一个 URL 分解,对于编程提供了相当大的方便。

【例 5-14】 改变 URL 示例。

```
<html>
<body>
<form>
   <input type = "button" value = "Google"
            onClick = "document.links[0].href = 'http://www.google.com' ">
   <input type = "button" value = "百度"
            onClick = "document.links[0].href = 'http://www.baidu.com' ">
   <input type = "button" value = "北大天网"
            onClick = "document.links[0].href = 'http://www.pku.edu.cn' ">
</form>
<a href = "javascript:alert('选择搜索引擎,然后单击下一步')">下一步</a>
</body>
</html>
```

<a href="javascript:alert('请按按钮选择下一步的目的地!')">下一步</a>是网页中定义的一个超链接,存储在 document.links[0].href 中。

我们知道,在<a>标记中, href 属性中应该是一个 URL,但是在这个句子中使用的是一种不常见的方式。其中,"javascript:"表示 JavaScript 语句前缀,后面为 JavaScript 语句。再如,<a href="http://www.google.com" onClick="alert('使用 Google 搜索引擎');return(true)">,注意,最后的 return(true) 语句命名这个 URL 真正被载入,当返回 false 时是不会真正载入 http://www.google.com 这个地址的。

这是因为,在 html 中,<a>标记可以同时设置 href 属性和 onclik 时间属性,<a>标记的执行顺序是先执行 onclick 的脚本,最后才进行 href 参数指定页面的跳转。因此,如果在 onclick 中返回 false,就可以中止<a>标签的工作流程,也就是不让页面跳转到 href 参数指定的页面。

例如:

```
<a href = '#' onclick = "location = 'aa.html';">aaa</a>
<a href = '#' onclick = "location = 'aa.html'; return false;">bbb</a>
```

在单击第一个超链接时,页面会显示没有转向到 aa.html。

对于上面的例子,当用户单击其中的一个按钮时,document.links[0].href 的值被改变,因此此时单击"下一步"超链接就转到相应的 URL。当鼠标在"下一步"超链接上经过时,浏览器窗口的状态栏内显示了这种 URL 的变化。

## 5.8 Web 交互

通过 JavaScript,可以完成 Web 的信息交互,在客户端完成动态的 Web 效果。主要的交互包括利用 form 对象,进行信息的输入,在帧中进行页面之间的控制。

### 5.8.1 使用 form 实现 Web 页面的信息交互

在 Web 页中,表单(form)是人机交互的主要手段。每一个<form>标记都在内存中创建一个 form 对象,通过 form 对象,可以直接访问 HTML 文档中的表单中的内容。Form

对象封装了相关的 HTML 代码：

```
<form name="表单名称" target="指定信息的提交窗口" action="接收窗体程序对应的 url"
method=信息数据传送方式(get|post) enctype ="窗体编码方式" >
```

### 1. form 对象的属性

在 HTML 中，表单是由若干控件(属性域)组成的，因此，一个 form 对象即是一个容器对象。form 对象的属性主要包括：elements、name、action、target、encoding、method 等。其中，elements 为元素数组，存储表单中的所有控件对象。其他几个均反映了<form>标记中的相应属性。对表单中的属性域，可以通过“表单名.elements[下标]”来访问，也可以通过“表单名.控件名”来访问，例如 myform.elements[0]、myform.account 等。

### 2. form 对象的方法

form 对象的方法只有一个，即 submit()方法，该方法主要功能就是实现 form 信息的提交。如提交 mytest 表单，则使用下列格式：

```
document.mytest.submit();
```

### 3. 访问 form 对象

在 JavaScript 中，访问窗体对象可由两种方法实现：

方法一：通过 form 名称访问

在<form>标记中，包含一个 name 属性，它对应了 from 对象的对象名，JavaScript 程序可以通过表单名来访问表单，例如：document.myForm()。

方法二：通过数组来访问 form

除了使用 form 对象名来访问页面中的表单外，还可以通过 document 对象的 forms 对象数组来访问 form 对象，因此可通过下列格式实现 form 对象的访问：

```
document.forms[0]、document.forms[1]、document.forms[2]…
```

需要说明的是，在 JavaScript 中要对 form 引用的条件是：必须先在页面中用<form>标记创建表单，并将定义表单部分放在引用之前。

### 4. 访问 form 中的元素

一个表单是由若干的表单控件组成的，这些控件包括：文本框(text)、单选钮(radio)、复选框(checkbox)、按钮(button)等。每一个控件，在内存中都创建相应的内存对象。在 JavaScript 中要访问这些基本元素，必须通过对应特定的 form 元素的数组下标或 form 元素名来实现。每一个元素主要是通过该元素的属性或方法来引用。其引用的基本格式为：

```
formName.elements[].propertyName | methodName
```

或者：

```
formName.elementName.propertyName | methodName
```

不同的 form 控件，对应的内存对象的属性和方法也不相同，但每一个内存对象包含的属性和方法均与其对应的 HTML 标记对应。例如，对于 text 内存对象，对应一个单行文本框输入控件。在 HTML 中，标记单行文本框 text 输入的一般形式是：

```
<input type = "text" name = " " value = " ">
```

因此，对应于每一个 text 对象，其包含的属性包括：name 属性、value 属性、defaultvalue 属性等。

对于每一个输入对象，通常还包含一组常用的方法，如：blur()方法，将当前焦点移到后台；select()方法，选择 text 框内的文本。包含的事件有：onFocus，当 text 获得焦点时，产生该事件；onBlur，从元素失去焦点时，产生该事件；onselect，当文字被选中，产生该事件；onchange，当元素值改变时，产生该事件。

**【例 5-15】** 表单控件的操作示例。

```
<html>
<head>
<script Language = "JavaScript">
function test()
{
      document.myForm.text1.value = "yyy";
      document.myForm.text1.select();
}
</script>
<head>
<body>
<form name = "myForm">
      <input type = "text" name = "text1" value = "xxx" >
</form>
<a href = "#" onclick = "test()">test</a>
</body>
</html>
```

单击网页中的超链接“test”，文本框的内容将赋值为“yyy”，并被选中。

### 5.8.2 使用 frame 实现更复杂的交互

框架可以将屏幕分割成不同的区域，每个区域有自己的 URL，通过 window 对象的 frames[]数组对象可以实现不同框架的访问。实际上框架对象本身也是一类窗口，它继承了窗口对象的所有特征，并拥有所有的属性和方法。

例如，下面是一个包含三个框架的框架网页。

```
<html>
<frameset rows = "20%,80%">
    <frame src = "frameA.htm">
    <frameset Cols = "200, *">
        <frame src = "frameB.htm">
        <frame src = "frameC.htm">
```

```
</frameset>
</frameset>
</html>
```

以上 HTML 标识将屏幕分成三个框架。先将窗口分成两行,之后再将第二行分成两列,其中,第一列占用 200 像素的固定宽度。

在前面我们介绍过使用 document.forms[]实现单一 form 中不同元素的访问。而要实现框架中多 form 的不同元素的访问,则必须使用 window 对象中的 frames 属性,frames 属性为数组对象,通过下标实现不同框架的访问。

例如:parent.frames[i].docuement.forms[j]

除了使用数组下标来访问 frame 和 form 外,还可以使用框架名和 form 名来实现各元素的访问,形式如下:

```
parent.frameName.document.formName.elementName;
```

# 5.9 使用 AJAX 技术

在 Web 应用蓬勃发展的今天,新的技术也不断出现。在客户端编程方面,继 DOM 之后,为了更新页面局部的需要,出现了 AJAX 技术,它开创了一种新的 JavaScript 编程的思路和方法。AJAX 技术一旦出现,就受到了开发人员的欢迎,并得到了迅速的发展,并被广泛应用于许多需要实时刷新的页面中。

## 5.9.1 AJAX 基础

AJAX 技术是由 Jesse James Garrett 于 2005 年 2 月在一篇文章中提出来的,是 Asynchronous JavaScript and XML(异步 JavaScript 和 XML)的缩写,AJAX 提供与服务器异步通信的能力,一个最简单的应用是无须刷新整个页面而在网页中更新一部分数据。当时,AJAX 只是一种设想,其灵感来源于在需要用 Flash 来实现一些网络应用的功能,但是,由于对 Flash 的不熟悉,尝试用传统的 Web 技术来达到 Flash 的效果,即用 JavaScript 和 XML 这两种传统的 Web 技术来实现,这就出现了 AJAX 的概念。现在,这个术语已经用来泛指所有允许浏览器在不刷新整个页面的情况下与服务器通信的技术。

什么是 AJAX 技术呢?AJAX 技术实际上是把 JavaScript、CSS、DOM 和 HTML 结合起来的一种新用法。这种结合并不是新概念,在动态 HTML 中,人们已经将这些技术结合在一起使用了,AJAX 技术的独到之处在于它在服务器端使用了异步(asynchronous)处理技术。

我们知道,Web 应用软件在运行时需要大量的页面,用户在网页上输入数据时,单击“提交”按钮,客户端浏览器则把这些信息发送到服务器端,服务器根据用户的操作发送一个新页面到客户端。例如,在一个登录页面中,除了登录表单,页面上通常还包含网站品牌信息、导航条和版权声明等。对于传统的登录页面,当用户提交表单后,服务器将比较用户在表单中输入的数据与数据库中保存的登录信息是否一致。如果用户输入的数据不正确,服

务器就把与原来相同的登录页面重发给用户，而这个页面与原来的页面相比可能只是多了“登录失败”的消息。为了这小小的改变必须要重新传输整个网页。用户每发出一个请求，整个网页就要被刷新一次，即页面的加载与用户的请求是同步的。

刷新整个页面除了带来较大的网络流量外，对于一些聊天类的网站，频繁的页面刷新必然会产生闪烁，影响用户的视觉体验。当采用 AJAX 技术后，情况将会大大改善，当用户提交表单后，如果登录失败，将不再刷新整个网页，而是仅仅在页面上增加了“登录失败”的消息文本，即 AJAX 技术可以实现网页的局部刷新，而页面上的所有没有被刷新的信息都是提交表单前页面的内容。这无论是对于服务器的 CPU 开销还是对网络的传输开销，无疑都减轻了不少压力，也避免了页面闪烁现象的发生。

总之，AJAX 是一种客户端实现方法，开发人员不需要学习一种新的语言，在大多数现代浏览器中都能使用。AJAX 不关心服务器是什么，使用 AJAX 技术，不必丢掉原先掌握的服务器端技术，它可以与 ASP、JSP、PHP 等服务端脚本交互。尽管存在一些很小的安全限制，网站设计者还是可以充分利用原有的知识，很容易地使用 AJAX 技术。

## 5.9.2　XMLHttpRequest 对象

AJAX 技术的组成元素涉及 JavaScript、CSS、DOM、XMLHTTP 和 XML 等内容，对于 JavaScript、CSS、DOM 和 XML，在前面的章节中都已经做了较全面的介绍，本节将详细介绍 XMLHTTP 相关对象的概念及使用方法。

### 1. XMLHttpRequest 对象

1999 年春，在 IE 5.0 中，增加了一个新的 ActiveX 控件，即 XMLHttpRequest 对象(XHR)，当时，这个对象主要是在 IE 中使用。从 Mozilla 1.0 和 Safari 1.2 开始，对 XHR 对象的支持开始普及。这个很少使用的对象和相关的基本概念甚至已经出现在 W3C 标准中，即 DOM Level 3 加载和保存规约(DOM Level 3 Load and Save Specification)。现在，特别是随着 Google Maps、Google Suggest、Gmail、Flickr、Netflix、A9 等应用变得越来越炙手可热，XHR 已经成为事实上的标准，是 AJAX 技术的核心组成部分。

XMLHttpRequest 对象位于客户端浏览器中，是用来实现网页与 Web 服务器之间异步通信的对象，通过它可以在不进行整个网页刷新的情况下完成向服务器发出请求、接收响应等工作。AJAX 技术工作机制如图 5-11 所示。

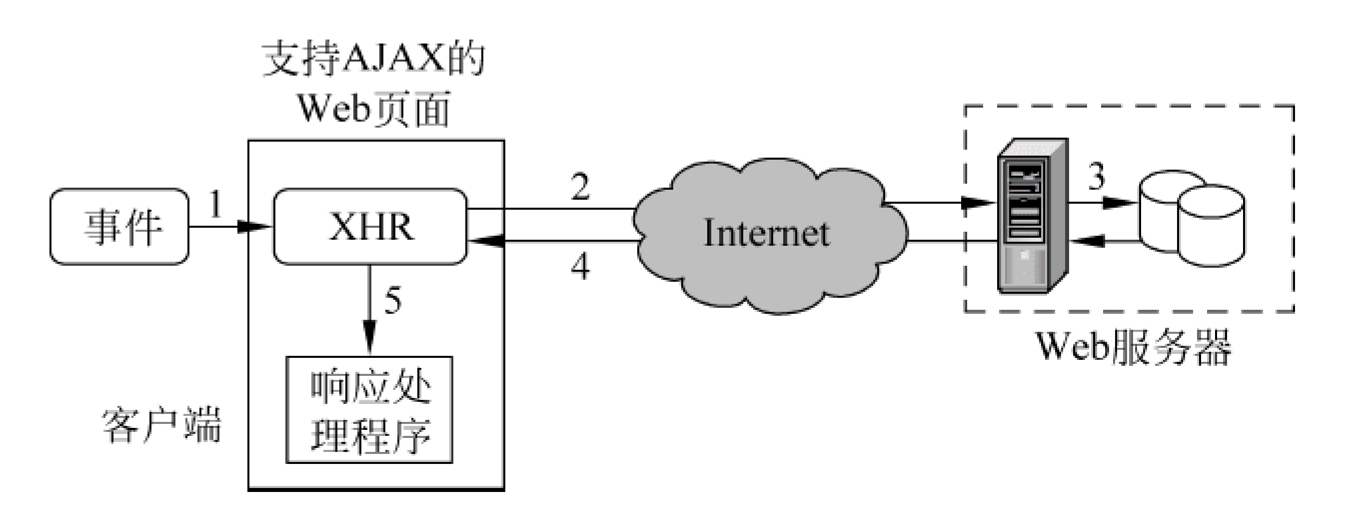



图 5-11　AJAX 技术工作机制

对于一个支持 AJAX 的 Web 页面来说，与服务器进行异步数据通信的过程如下：

① 当要求进行与服务器异步通信的某一事件发生时，事件处理程序将调用 XHR，设置该对象相关的属性参数。

② 由 XHR 向服务器发出请求，请求通过 Internert 发送到 Web 服务器。当服务器收到请求后，运行指定的服务器端程序。

③ 服务器端程序执行中如果包含数据库访问的命令，则连接数据库服务器完成数据库的相关操作，结果返回到 Web 服务器。

④ Web 服务器将响应信息通过 Internet 发回到客户端，浏览器将这些响应信息交给 Web 页面的 XHR 对象。

⑤ 启动相应的处理程序，通过该文档的 DOM 模型完成页面的更新工作。

AJAX 技术等于在客户端页面和服务器之间安插了一个中转站，JavaScript 脚本先把请求从页面发送给这个中转站，再由这个中转站把请求发给服务器；服务器对请求处理后，先把响应发给中转站，再由中转站将响应转发给页面，客户端页面的脚本程序就可以根据收到的数据进行网页的更新工作。这个中转站的角色就由客户端的 XMLHttpRequest 对象承担。在客户端响应数据处理程序一般为 XHR 对象状态改变的事件处理函数，具体的网页更新工作可通过当前页面的 DOM 模型完成。

### 2. 创建 XMLHttpRequest 对象

在使用 XMLHttpRequest 对象之前，必须先创建一个 XMLHttpRequest 对象。由于 XMLHttpRequest 不是一个 W3C 标准，所以需要针对不同的浏览器采用不同方法创建 XMLHttpRequest 的实例。Internet Explorer 把 XMLHttpRequest 实现为一个 ActiveX 对象，其他浏览器(如 Firefox、Safari 和 Opera)把它实现为一个本地 JavaScript 对象。由于存在这些差别，JavaScript 代码中必须包含有关的逻辑，从而使用 ActiveX 技术或者使用本地 JavaScript 对象技术来创建 XMLHttpRequest 的一个实例。

在这里为了明确该如何创建 XMLHttpRequest 对象的实例，并不需要那么详细地编写代码来区别浏览器类型。你要做的只是检查浏览器是否提供对 ActiveX 对象的支持。如果浏览器支持 ActiveX 对象，就可以使用 ActiveX 来创建 XMLHttpRequest 对象。否则，就要使用本地 JavaScript 对象技术来创建。可以通过以下函数创建 XMLHttpRequest 对象：

```
var xmlHttp;
function createXMLHttpRequest()
{   if(window.ActiveXObject)
        xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
    else if (window.XMLHttpRequest)
        xmlHttp = new XMLHttpRequest();
}
```

在这里，变量 xmlHttp 保存了 XMLHttpRequest 对象新创建的一个实例，如果创建失败，xmlHttp 的值将为 null。通过该对象的属性和方法，可以很容易地与服务器实现请求的发送和响应的接收。

### 3. XMLHttpRequest 对象的属性和方法

XMLHttpRequest 对象常用属性见表 5-14。

表 5-14　XMLHttpRequest 常用属性

| 属　　性 | 描　　述 |
|---|---|
| onreadystatechange | 状态改变时发生的事件,可以将它与一个 JavaScript 函数绑定 |
| readyState | 请求的状态。有 5 个可能的取值(0:未初始化;1:正在加载;2:已加载;3:交互中;4:完成) |
| responseText | 服务器的响应,表示为一个串 |
| responseXML | 服务器的响应,表示为 XML。可以解析为一个 DOM 对象 |
| status | 服务器的 HTTP 状态码。例如,200:OK,404:Not Found,等等 |
| statusText | HTTP 状态码的相应文本。例如,OK 或 Not Found(未找到)等 |

标准 XMLHttpRequest 对象的方法见表 5-15。

表 5-15　XMLHttpRequest 对象常用方法

| 方　　法 | 描　　述 |
|---|---|
| open("method", "URL", "asynch") | 建立对服务器的请求,即设置对应服务器端的异步通信程序,URL 为服务器端程序的 URL |
| send(content) | 向服务器发送请求 |
| setRequestHeader("header", "value") | 把指定首部设置为所提供的值。在设置任何首部之前必须先调用 open() |
| abort() | 停止当前请求 |
| getAllResponseHeaders() | 把 HTTP 请求的所有响应首部作为键/值对返回 |
| getResponseHeader("header") | 返回指定首部的串值 |

下面详细介绍这几个方法的应用。

① void open(string method, string url[,boolean asynch][,string username][, string password])

建立对服务器的调用,这是初始化一个请求的纯脚本方法,包含两个必要的参数和 3 个可选参数。其中:

method:必选参数,提供调用的特定方法(GET、POST 或 PUT)。

url:必选参数,所调用资源的 URL,即服务器端处理程序。

asynch:必选参数,指示这个调用是异步的还是同步的。默认值为 true,如果为 false,处理就会等待,直到从服务器返回响应为止。

username:可选参数,表示用户名,用于身份验证。

password:可选参数,表示用户密码,用于身份验证。

② void send([content])

向服务器发出请求。如果请求声明为异步的,这个方法就会立即返回,否则它会等待直到接收到响应为止。可选参数可以是 DOM 对象的实例、输入流,或者串。传入这个方法的内容会作为请求体的一部分发送。

③ void setRequestHeader(string header, string value)

HTTP 请求中一个给定的首部设置值。它有两个参数:header 表示要设置的首部;value 表示首部的值。需要说明,这个方法必须在调用 open()之后才能调用。

④ void abort()

停止请求的方法。

⑤ string getAllResponseHeaders()

返回所有响应的 Http 头,首部包括 Content-Length、Date 和 URI。

⑥ string getResponseHeader(string header)

返回指定的首部值。

由于 AJAX 技术涉及客户端和服务端编程两个方面,我们将在下一章的综合举例中,通过聊天程序的实现,详细介绍 AJAX 技术的应用。

## 5.10 综合举例

Web 客户端的编程比较简单,它不需要特别的编译和运行环境,只要有一个浏览器就可以了。但是,要编写高质量的客户端脚本并不容易,这取决于您大量的实践经验,也来源于用户的需求。没有需求,就不会有深入的编程体验,更无法把一个工具学精。为此,在本章的最后,我们给出了三个在实际 Web 开发中常用的功能,分别来综合说明 JavaScript 中的窗口控制、图层技术和 HTML DOM 文档的操作,帮助建立总体的 Web 客户端编程思想。也相信这三个应用的代码,对用户会有很好的借鉴作用。

### 5.10.1 一个 Web 课件框架

在 E-learning 中,网络课件是重要的教学资源,一种简单易用的课件界面对于 E-learning 系统有着重要的作用。图 5-12 是我们的 Genaral Self-learning 网络教学平台 GSL 中的知识单元学习界面。

<table>
<tr><td colspan="2">一区(图片区)</td></tr>
<tr><td>二区<br>(黑板区)</td><td>三区<br>(讲解区)</td></tr>
<tr><td>四区(状态区)<br>显示在线人数和在线教师图标</td><td>五区(导航区)</td></tr>
</table>

图 5-12 Web 课件界面

一区显示课程的标题图片。二区为黑板区,在黑板区中主要展示课程中的插图、动画、视频、程序、公式推导、定理证明等教师在课堂教学中在黑板上书写的内容。三区为教案讲解区,主要是文字解说,对应教师在课堂教学中的讲解。黑板区与教案讲解区紧密配合,二区与三区配合完成整门课程的教学内容设计,通过标准的稿本创作规范由教师完成。四区为状态区,是学生和系统在线交互的主要入口,显示在线学生数量、在线辅导教师以及供学

生提问的帮助入口。五区为导航区，完成默认的页面导航，包括：目录、知识点、上一页、下一页、实验、练习题等超级链接，实现每一章学习内容的交互。

下面我们将利用 JavaScript 脚本程序来完成上述界面的设计，它包括了 JavaScript 中内嵌对象、框架、表单、参数传递、包含等各种各样的 JavaScript 编程技术。

### 1. 建立 Web 应用中的一个页面——初始框架

根据问题的需求，我们设计 Web 应用的总体框架。根据图 5-12，在这个 Web 应用中，第一个网页应该是一个框架网页，我们将主调文档的文件命名为 myframes. htm。

用 FrontPage 工具完成 myframes. htm 框架网页的设计，代码如下：

主调文档 myframes. htm 内容：

```
<html>
<head>
<title>GSL 网络课程</title>
<script>
  function maxwindow()
  {
    window.moveTo(0,0);
    window.resizeTo(screen.availWidth,screen.availHeight);
  }
</script>
</head>
<frameset rows = "100, * ,60" framespacing = "0" border = "0" frameborder = "0"
          onLoad = "maxwindow()">
  <frame name = "topbanner" scrolling = "no" noresize src = "mybanner.htm">
  <frameset cols = " * ,300">
    <frame name = "frameA" target = " _self" scrolling = "auto" src = "myA.htm" >
    <frame name = "frameB" target = "frameA" scrolling = "auto" src = "myB.htm" >
  </frameset>
  <frameset cols = " * ,300">
    <frame name = "frameC" scrolling = "no" src = "myC.htm">
    <frame name = "frameD" scrolling = "auto"
           src = "myD.htm chapter = ch01&pagenums = 26&page = 0" >
  </frameset>
</frameset>
</html>
```

整个窗口分成五个帧，帧的名字分别是 topbanner、frameA、frameB、frameC 和 frameD。其中，topbanner 帧主要负责显示标题图片；frameA 帧用于显示知识单元所对应的图标、动画、公式等相关学习对象；frameB 中显示知识单元具体的学习内容；frameC 帧为状态显示，与其他帧没有超链接关系；frameD 为导航帧，它影响 frameB 的显示内容。

注意，在 HTML 中，top 是一个保留字，最好不要使用 top 作为帧的名字，否则在调用时可能出错。为了使得浏览器窗口发生变化的时候，fameA、frameC 和 frameD 帧不发生变化，我们使用了帧的绝对像素值，而不是百分比。

### 2. topbanner 帧的设计

topbanner 帧的设计显示一幅图片或 Flash 动画，我们指定它的源文件是 mybanner.

htm,这是最为简单的一个网页,只要设置一幅背景图片就可以了。或者根据帧的尺寸大小插入一个 Flash 动画。一个简单的 mybanner. htm 文档内容如下:

```
<html>
<head>
<meta http-equiv = "Content-Type" content = "text/html; charset = gb2312">
<title>Top banner</title>
</head>
<body background = "images/coursebanner.jpg">
</body>
</html>
```

### 3. frmaeC 帧的设计

在 frameC 帧,我们主要用于设计系统的状态,它对应了一个 myC. htm 网页,可能是显示目前的在线人数,这里先不考虑。

### 4. frmaeD 帧的设计

frameD 帧是我们的页面导航区,为简单起见,包括三个图片超链接:目录、上一页、下一页,每一个超链接对应不同的显示状态,是相对复杂的一个网页。

MyD. htm 文档内容如下:

```
<html>
<head>
<meta http-equiv = "Content-Type" content = "text/html; charset = gb2312" >
<script src = "QueryString.js" ></script>
<script src = "indexdatas.js" ></script>
<script language = "JavaScript">
    //(1)index 按钮,对应两个图片,分别用于当鼠标置于按钮上,或离开时的显示
    image1on = new Image();
    image1on.src = "images/indexov.gif";
    image1off = new Image();
    image1off.src = "images/indexup.gif";
    //(2)up 按钮,对应三个图片,分别用于当鼠标置于按钮上、离开时或当前页是第一页时的灰化
    //的显示
    image2on = new Image();
    image2on.src = "images/backov.gif";
    image2off = new Image();
    image2off.src = "images/backup.gif";
    image2dim = new Image();
    image2dim.src = "images/backdim.gif";
    //(3)next 按钮,对应三个图片,分别用于当鼠标置于按钮上、离开时或当前页是最后一页时的
    //灰化的显示
    image3on = new Image();
    image3on.src = "images/nextov.gif";
    image3off = new Image();
    image3off.src = "images/nextup.gif";
    image3dim = new Image();
    image3dim.src = "images/nextdim.gif";
```

```
// 初始化按钮的显示,其中 up 按钮显示为灰化
function myButtonInit()
{
    var mynum = parseInt(mypagenums,10);
    document['image2'].src = image2dim.src;
    if (mynum <= 1)
        document['image3'].src = image3dim.src;
    }
    // 改变按钮的显示图片
    function showPic()
    {
        if (document.images && PageNums > 1)
            for (var i = 0; i < showPic.arguments.length; i + = 2)
            {
              if (showPic.arguments[i] =='image3' && CurrentPage ==PageNums)
                  return;
            else if (showPic.arguments[i] =='image2' && CurrentPage ==1)
                    return;
                else
                document[showPic.arguments[i]].src = eval(showPic.arguments[i + 1] +
".src");
              }
    }
    // 打开一个索引页面,显示课程的三级目录,每个目录对应一个超链接
    function openIndexwin()
     {
       if (indexwin ++"" || indexwin.closed)
         {  // 书本的全部三级目录索引页面
            myurl = "indexframes.htm";
           indexwin = window.open("indexframes.htm","IndexWin","");
           //window.open("indexframes.htm","IndexWin","")
            //popup.location = myurl;
         }
          else
             indexwin.focus();
    }
    // 根据章和当前页,获得当前页对应的网页文件的文件名代码,代码数组存储在文件
    // indexdatas.js 中
    function getPagecode(cha,thePage)
    {
      var pagecode;
      switch (cha) {
            case "ch01":
                  pagecode = PagesIDs01[thePage];
                  break;
            case "ch02":
                  pagecode = PagesIDs02[thePage];
                  break;
            //……
          }
          return pagecode;
```

```
        }
        // 在 frameA 和 frameB,分别显示下一知识单元对应的 HTML 文件
        function NextPage()
        {
          if (CurrentPage ==PageNums - 2)
                  return;
          if ((CurrentPage  + 1) ==PageNums - 2)
              document['image3'].src  =  image3dim.src;
          CurrentPage++;
          pagecode  =  getPagecode(mycha,CurrentPage);
          NextPageURLa  =  mycha  +  "/"  +  pagecode  + "a" + ".htm";
          NextPageURLb  =  mycha  +  "/"  +  pagecode  + "b" + ".htm";
          eval('parent.frames.frameA.location.href = NextPageURLa');
          eval('parent.frames.frameB.location.href = NextPageURLb');
          }
          // 在 frameA 和 frameB,分别显示上一知识单元对应的 HTML 文件
          function PrevPage()
          {
            if (CurrentPage ==0)
                return;
            if (CurrentPage ==1)
                document['image2'].src  =  image2dim.src;
            CurrentPage--;
            pagecode  =  getPagecode (mycha,CurrentPage);
            BackPageURLa  =  mycha  +  "/"  +  pagecode  + "a" + ".htm";
             BackPageURLb  =  mycha  +  "/"  +  pagecode  + "b" + ".htm";
            eval('parent.frames.frameA.location.href = BackPageURLa');
            eval('parent.frames.frameB.location.href = BackPageURLb');
        }
</script>
</head>
<body leftmargin = "1"  topmargin = "1"  bgcolor = "#000000">
<script language = JavaScript>
    var Request = new QueryString();            // 创建参数对象实例,定义在 QueryString.js 中
    mycha         = Request["chapter"];         // 章编号
    mypagenums  =  Request["pagenums"];         // 当前章页数
    mypage        = Request["page"];            // 当前页
    var PageNums    =  parseInt(mypagenums,10);
    var CurrentPage =  parseInt(mypage,10);
    var indexwin = "";                          //索引窗口
</script>
<div align = "center">
    <center>
    <table border = "0"  cellpadding = "0"  width = "100%"  height = "100%">
      <tr>
        <td align = "center"  width = "100%">
          <table bgcolor = "#000000"  border = "0"  cellspacing = "0"  cellpadding = "0">
            <tr>
              <td><a href = "javascript:openIndexwin()"  target = "_self"
                        onMouseOver = "showPic('image1', 'image1on')"
                        onMouseOut = "showPic('image1', 'image1off')">
```

```
                <img name = "image1" src = "images/indexup.gif" ></a>
            </td>
            <td><a href = " javascript:PrevPage()"
                onMouseOver = "showPic('image2', 'image2on')"
                onMouseOut = "showPic('image2', 'image2off')">
                <img name = "image2" src = "images/backdim.gif" ></a>
            </td>
            <td><a href = " javascript:NextPage()"
                onMouseOver = "showPic('image3', 'image3on')"
                onMouseOut = "showPic('image3', 'image3off')">
                <img name = "image3" src = "images/nextup.gif" ></a>
            </td>
          </tr>
        </table>
      </td>
    </tr>
  </table>
  </center>
</div>
<script language = "JavaScript">
  myButtonInit();
</script>
</body>
</html>
```

myD.htm 显示结果如图 5-13 所示。

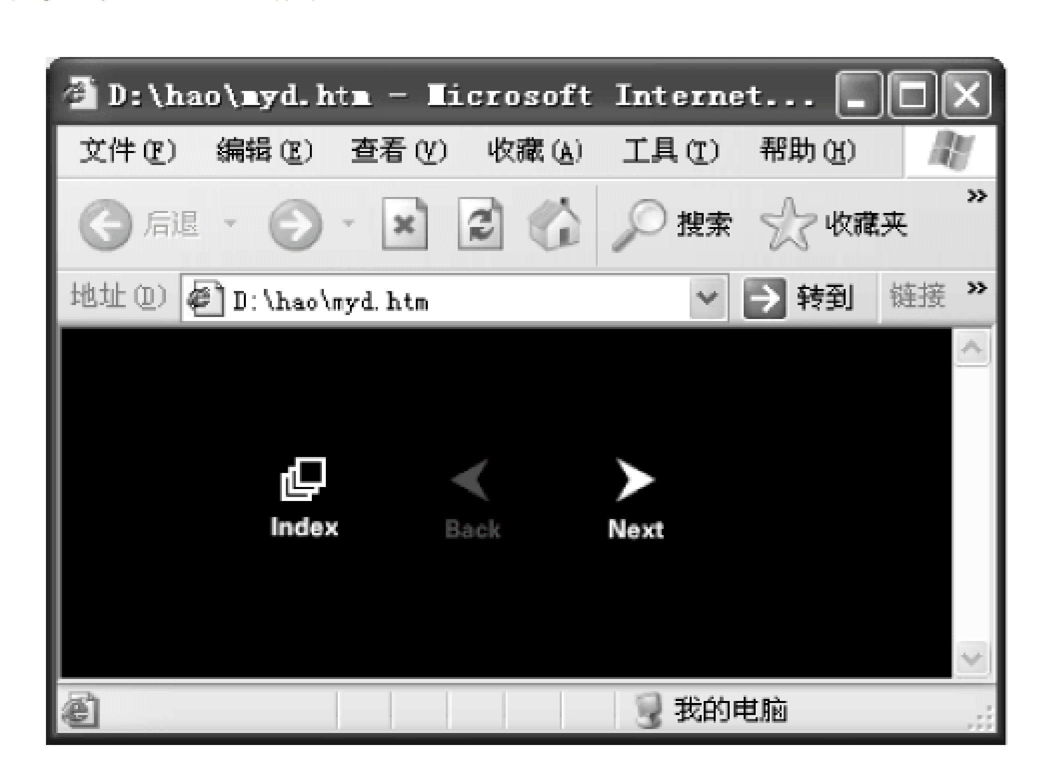



图 5-13　frameD 帧的显示

说明：

(1) 在 myD.htm 中，我们看到每一个图片超链接的 href 属性的值写成了类似下面的形式：href="javascript: openIndexwin()"，其中的前缀“javascript: ”是不可缺少的，每一个<a>标记默认打开一个窗口，因为需要的窗口是要在 openIndexwin()中打开的，因此在<a>标记中，设置属性 target 为“_self”，用以避免单击该<a>时，打开两个窗口，一个是<a>对应的窗口，一个则为函数内部用 open 打开的窗口。

此外，在<a>标记中，可以同时设置 href 属性和 onclick 属性。如果设置了 onclick 属性，浏览器将首先执行 onclick 中的代码，如果有返回语句 return false，则不再执行 href 中

的地址转移,否则,执行完 onclick 中的代码后,将转移到 href 参数中设定的 URL。

例如,大家可以验证下面的 HTML 语句:

```
<a href = "#" onclick = "Javascript:test();return false;">test</a>
```

取消 return false,再尝试一次,看看结果有何不同。

(2) 在 myD.htm 中,如果要修改 frameA 和 frameB 中的 HTML,也可以不使用 eval 函数,而直接使用类似下列的代码:

```
window.parent.parent.frames['frameA'].location = "url";
```

(3) 对于一个图片或一段文本可以设置水平居中,如果要使一个图片做到在网页内水平和垂直居中,如何实现呢? 方法是将内容插入到一个 1×1 的表格中,指定表格属性设置 width="100%",height="100%",然后在单元格中插入相应的内容即可。

(4) 包含文件

在 myD.htm 文档中,用到了两个包含文件:QueryString.js 和 indexdatas.js。其中 QueryString.js 定义了一个参数类,用于获得当前页面的参数,在本例子中,传递到 myD.htm 页面的有三个参数,分别是:“chapter”(章编号)、“pagenums”(当前章的页数)和“page”(当前页)。

QueryString.js 文档内容如下:

```
function QueryString()
{ //构造参数对象并初始化
    var name,value,i;
    var str = location.href;                //获得浏览器地址栏 URL 串,本例中的形式为:
                                            // myD.htm?chapter = ch01&pagenums = 26&page = 0
    var num = str.indexOf("?")
    str = str.substr(num + 1);              //截取“?”后面的参数串
    var arrtmp = str.split("&");            //将各参数分离形成参数数组
    for(i = 0;i < arrtmp.length;i++)
    {
      num = arrtmp[i].indexOf(" = ");
      if(num>0)
      {
        name = arrtmp[i].substring(0,num);  //取得参数名称
        value = arrtmp[i].substr(num + 1);  //取得参数值
        this[name] = value;                 //定义对象属性并初始化
      }
    }
}
```

包含文件 indexdatas.js 定义了整个 Web 应用所有网页用到的全局变量,内容如下:

```
var PageNums01  =  26;              // 第 1 章知识单元数,一个知识单元对应一个 Web 页
var PagesIDs01  =  new Array(26);   // 存储每一页的 ID = p + 节 + 小节,节和小节各占两位
PagesIDs01[0]  =  "p00";            // 本章的目录页
PagesIDs01[1]  =  "p0101";          // 第 1.1.1 小节
PagesIDs01[2]  =  "p0102";
```

```
PagesIDs01[3] = "p0103";
//…
PagesIDs01[25] = "p0702";
// 以下是第二章、第三章等其他章的数据，和第一章类似。
```

（5）函数调用

在一个帧中打开了许多页面，只要页面没有关闭，则其中的函数也都在内存中，因此，这些页面中定义的函数可以互相调用。因为页面中定义的函数都属于这个页面对应的 window 对象的成员，因此，可以通过相应的 window 对象来调用。

例如，在本例中的 myD.htm 的函数定义中，可以调用 topbanner 中定义的函数 f1()。因为根据 myframes 中的帧结构定义，在 frameD 中的页面 myD.htm 要定位到 topbanner 帧窗口中的页面，需要一个 window 对象的复杂引用，具体的调用方法是：

```
window.parent.parent.frames['topbanner'].f1();
```

此外，也可以使用在另一个打开的页面中定义的全局变量，使用方法和上述方法类似，这些变量都是窗口对象 window 的成员。

### 5. 目录索引页设计

在 frameD 帧中，有一个"index"按钮，单击该按钮，将打开一个新的窗口，显示一门课程中的课程目录，通过该目录，学生可以进入任何一章、任何一个小节。此时，在 frameA 和 frameB 中显示该小节的内容。

（1）目录索引页 indexframes.htm 设计成一个框架网页，如图 5-14 所示。

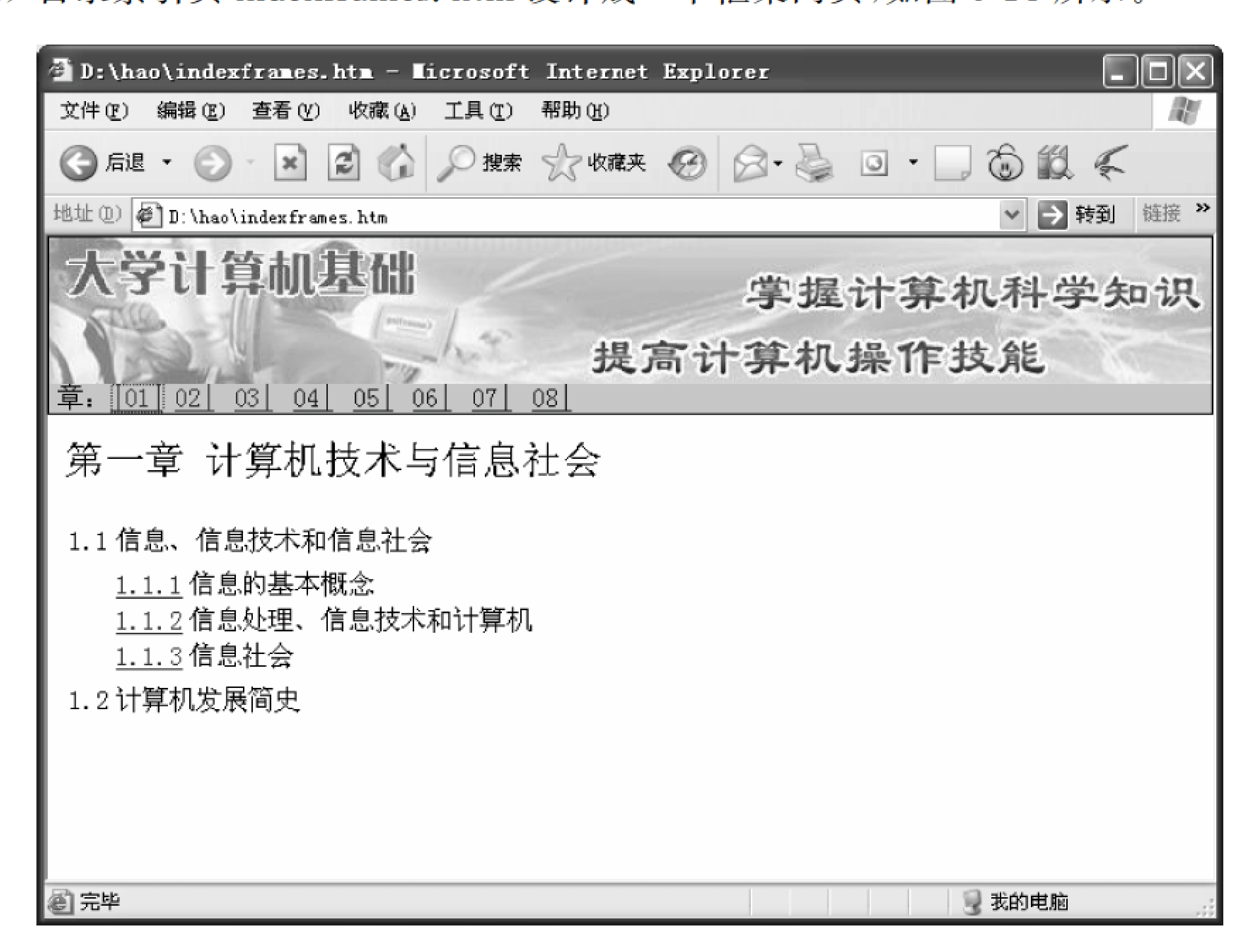



图 5-14　目录索引页

indexframes.htm 框架网页包含了上下两个帧，上面显示菜单(myindexmenu.htm)，下面是每一章的目录(每章一个文件/chxx/pindex.htm)。

indexframes.htm 代码如下：

```
<html>
<head>
<meta http-equiv = "Content-Type" content = "text/html; charset = gb2312" >
<script>
var activeindex = null;
function showLayer(layer)
{
eval(activeindex + ".hide()");
activeindex = layer;
eval(layer + ".show()");
}
function hideLayer(layer)
{
eval(layer + ".hide()");
}
</script>
</head>
<frameset frameborder = "0" border = "0" framespacing = "0" rows = "110, * ">
    <frame name = "indexmenu" src = "courseindex.htm" target = "_self" scrolling = "no">
    <frame name = "indexcontent" src = "pleaseselcha.htm" target = "main">
</frameset>
</html>
```

(2) 菜单页面 courseindex.htm 的内容如下：

```
<html>
<head>
<meta http-equiv = "Content-Type" content = "text/html; charset = gb2312" >
<script language = "JavaScript">
function chapterIndex(chapter)
{
     str = "pindex.htm"; // 每一章的目录页,分别存在 ch01、ch02……文件夹下
      indexPageURL = chapter + "/" + str;
      window.parent.frames['indexcontent'].location = indexPageURL;
}
</script>
<base target = "_self">
</head>
<body leftmargin = "1" topmargin = "1" bgcolor = "#000000">
    <div align = "center">
    <table bgcolor = "#000000" border = "0" cellspacing = "0" cellpadding = "0" width = "100%">
        <tr>
            <td width = "540" height = "90" colspan = 9 background = "images/coursebanner.jpg">
            </td>
        </tr>
        <tr bgcolor = "#FFCC00">
            <td width = "40" align = "right">章:</td>
            <td width = "40"><a href = "javascript:chapterIndex('ch01');">|01|</a></td>
```

```
        <td width="40"><a href="javascript:chapterIndex('ch02');">02|</a></td>
        <td width="40"><a href="javascript:chapterIndex('ch03');">03|</a></td>
        <td width="40"><a href="javascript:chapterIndex('ch04');">04|</a></td>
        <td width="40"><a href="javascript:chapterIndex('ch05');">05|</a></td>
        <td width="40"><a href="javascript:chapterIndex('ch06');">06|</a></td>
        <td width="40"><a href="javascript:chapterIndex('ch07');">07|</a></td>
        <td width="550"><a href="javascript:chapterIndex('ch08');">08|</a></td>
      </tr>
    </table>
    </div>
</body>
</html>
```

### 6. 每一章的目录页

在目录窗口帧中，单击某一章，在 indexcontent 帧中将显示所选章的目录。每一章的目录文件名为 pindex.htm，分别存储在 ch01、ch02……子文件夹中。该文件含有大量的超链接，最终通过这些超链接来刷新 frameA 和 frameB 中的内容。

pindex.htm 代码如下：

```
<html>
<head>
<meta http-equiv="Content-Language" content="gb2312">
<title>pindex01</title>
<script src="../indexdatas.js" ></script>
<script language="JavaScript">
function GoContentPage(selstr)
{
    Chapter    = selstr.substring(0,2);
    Section    = selstr.substring(2,4);
    Subsection = selstr.substring(4,6);
    str = "p" + selstr.substring(2,6);
    CurrentPage = 0;
    //为简单起见，只考虑第一章的情况，PageNums01 为第一章的页面数，存储在 indexdata.js
    //文件中
    while (CurrentPage<PageNums01)
    {
        if (PagesIDs01[CurrentPage] == str)
            break;
        CurrentPage ++;
    }
    if (CurrentPage == PageNums01)
       CurrentPage = 0;
    NewPageURLa = "../ch" + Chapter + "/" + PagesIDs01[CurrentPage] + "a" + ".htm";
    NewPageURLb = "../ch" + Chapter + "/" + PagesIDs01[CurrentPage] + "b" + ".htm";

    // document.writeln(NewPageURLa);
    if (window.parent.opener && !window.parent.opener.closed)
```

```
        {
            window.parent.opener.parent.frames['frameA'].location.href = NewPageURLa;
            window.parent.opener.parent.frames['frameB'].location.href = NewPageURLb;
        }
        // 刷新 frameD 帧的内容,使得其中的数据和索引页的选择同步
        newurl = "../myD.htm chapter=ch01&pagenums=26&page=" + CurrentPage;
        window.parent.opener.parent.frames['frameD'].location = newurl;
        top.blur();
    }
    </script>
    </head>
    <body>
    <p><font size="5">第一章 计算机技术与信息社会</font></p>
    <div align="center">
        <center>
          <table border="0" width="100%">
          <tr height="30">
            <td >1.1</td>
            <td colspan="2" align="left">信息、信息技术和信息社会</td>
          </tr>
          <tr>
            <td width="3%"></td>
            <td width="5%"><a href = "#" onClick="GoContentPage('010101');">1.1.1</a>
            </td>
            <td align="left" width="90%">信息的基本概念</td>
          </tr>
          <tr>
            <td width="3%"></td>
            <td width="5%"><a href = "#" onClick="GoContentPage('010102');">1.1.2</a>
            </td>
            <td align="left" width="90%">信息处理、信息技术和计算机</td>
        </tr>
        <tr>
            <td width="3%"></td>
            <td width="5%"><a href = "#" onClick="GoContentPage('010103');">1.1.3</a>
            </td>
            <td align="left" width="90%">信息社会</td>
        </tr>
        <tr height="30">
              <td >1.2</td>
              <td colspan="2" align="left">计算机发展简史</td>
        </tr>
    </table>
    </center>
    </div>
    </body>
    </html>
```

当上述代码完成后,双击 myframes.htm 文件,在浏览器中显示系统主界面,单击 Index、back 和 next 按钮,可以检验系统的运行效果。系统界面如图 5-15 所示。




图 5-15　最终完成的 GSL 网络课程界面

#### 7. 总结

上述的代码很长，从中我们可以体会到，Web 应用的客户端编程还是非常复杂的，特别是在不同的帧之间的控制，以及页面之间的参数传递和函数调用。这需要对 JavaScript 内部对象、浏览器 BOM 模型和 HTML DOM 模型有非常清楚的理解。

同时，一个功能可能有多种实现方法，上述代码中，也尽量地反映出这种多样性。只要经过认真的积累，定会编写出精彩的 Web 应用。

### 5.10.2　一个文本文档批注系统

对于写在纸面上的东西，我们很容易再对内容进行修改或添加说明。如果是电子文档，如何完成上述的功能呢？如果是带有格式的文档，如 Word 文档，可以用不同的颜色，通过“审阅”工具栏、插入“批注”等方式对文档进行说明。

对于文本文档，由于没有格式，因此无法用类似 Word 的方法来实现对文档的说明。因此，可以利用 HTML 和 JavaScript 技术，开发一个文本文档的批注系统。

#### 1. 登录界面设计

首先设计一个登录界面，不同的角色可以具有批注权限。设计的登录界面如图 5-16 所示。




图 5-16 登录界面

对应的 HTML 文档 mylogin. htm 内容如下：

```
<html>
<head>
<title>postil</title>
<link href = "mypub.css" rel = "stylesheet" type = "text/css">
<script language = "javascript" src = "myglobal.js"></script>
<script language = "javascript">
function login()
{
  var name  =  document.all.userName.value;
  if( isEmpty( name ) )
  {
     alert( "用户名不能为空!" );
     document.all.userName.focus();
     return;
  }
  var r  =  document.all.userRole.value;
  if( r =="teacher" )
     window.open( "fileload.htm" , "_self" );
}
</script>
</head>
<body onload = "javascript:document.all.userName.focus();" bgcolor = "#CCCCCC" >
<div align = "center">
  <center>
  <table width = "100%" height = "100%" border = "0" cellpadding = "0">
  <tr>
  <td width = "100%">
  <table width = "300" align = "center" border = "2" cellspacing = "0" cellpadding = "0" >
       <tr height = 50><td align = "center">文本文档批注系统</td></tr>
       <tr><td>用户名 <input type = "text" name = userName></td></tr>
       <tr>
```

```
                <td>角   色  
                    <select name = userRole>
                        <option value = teacher>教师</option>
                        <option value = student>学生</option>
                    </select>
                </td>
            </tr>
            <tr><td><input type = "button" value = "登录" onclick = "login()"></td></tr>
        </table>
        </td>
        </tr>
        </table>
        </center>
    </div>
    </body>
    </html>
```

说明：在上述的文档中，引用了一个.css 文件 mypub.css，它定义了应用中的样式。同时，还用到了一个包含文件 myglobal.js，定义了各个 HTML 文档用到的公用函数和全局变量。

（1）mypub.css 样式文件内容

```
body {
     font - family: "宋体"; font - size: 9pt
     scrollbar - highlight - color: rgb(92,149,255);
     scrollbar - 3dlight - color: rgb(92,149,255);
     scrollbar - arrow - color: rgb(70,255,229);
     scrollbar - track - color: rgb(92,149,255);
     scrollbar - darkshadow - color: rgb(92,149,255);
     scrollbar - base - color: rgb(88,120,255);
}
table {font - family: "宋体"; font - size: 9pt}
td {font - family: "verdana"; font - size: 9pt}
input {
  border:1 solid black;font - size: 9pt; height: 18px; line - height: normal
}
textarea {font - family: "宋体"; font - size: 9pt}
select {font - family: "宋体"; font - size: 9pt}
a {font - family: "宋体"; font - size: 9pt;color: red;text - decoration:none}
a:hover {color: #cc0000}
p {font - family: "courier","verdana", "宋体"; font - size: 9pt }
p.foot {font - family:"宋体";font - size:9pt}
```

在上述样式表定义中，将<a>定义为红色显示，以表明该文本处包含用户批注。

（2）全局包含文件 myglobal.js 内容

```
function isEmpty( value )
{
    var temp = value;
    while (temp.indexOf( " " , 0 ) ! = -1 )
          temp = temp.replace( " " , "" );
```

```
        if (temp =="" ) return true;
        else    return false;
    }
```

### 2. 选择被批注文件并显示

登录后，如果是教师角色，则在当前窗口打开一个 fileload.htm 文档，它是选择被批注文件、进行批注的主界面。界面设计如图 5-17 所示。




图 5-17　设计中的文本批注主界面

我们将 fileload.htm 文档的内容分成两个部分：第一部分是选择一个要批注的文本文件，然后显示；第二部分是对文件进行批注。代码清单如下：

```
<html>
<head>
<title>file process</title>
<link href = "mypub.css" type = "text/css" rel = "stylesheet">
<script language = "javascript" src = "myglobal.js"></script>
<script language = "javascript">
   var fileName, savePath, saveFolder, saveName;
   var fileObj  =  new ActiveXObject("Scripting.FileSystemObject");
   var forReading  =  1;
   var forWriting  =  2;
   var forAppending  =  8;
 ////////////////////////////////////////////////////////////////////////////////
 //选择要批注的文件,并打开显示
 //
 function readFile()
  {
    fileName  =  document.all.homework.value;
    // 根据打开的文件,创建与打开文件同名的文件夹,以及生成要保存的同名的.htm 文件名
    var str  =  fileName;
    var name  =  fileName.substring( str.lastIndexOf( "\\" )  +  1 );
    saveName  =  name.substring(0 ,name.lastIndexOf( "." ))  +  ".htm";
    saveFolder  =  saveName.substring(0 , saveName.lastIndexOf( "." ) );
    if (!fileObj.FolderExists( "c:\\"  +  saveFolder ) )
           var folder  =  fileObj.CreateFolder( "c:\\"  +  saveFolder );
    savePath  =  "c:\\"  +  saveFolder  +  "\\" ;
    // 如果该文档已经批注过,则显示已经批注的内容
     if (fileObj.FileExists(savePath  +  saveName))
     {
```

```
        desFile = fileObj.OpenTextFile(savePath + saveName,forReading);
        fileContents = desFile.readAll();
        desFile.close();
        document.all.t.rows(2).cells(0).innerHTML = fileContents;
        //显示下面隐藏的表格(带有批语的文本框)
        document.all.t1.style.visibility = "visible";
        document.all.annotation.focus();
        return;
        }
      //打开并显示选择的文本文件
      var textFile = fileObj.OpenTextFile(fileName,forReading);
      if (textFile.AtEndOfStream)
      {
        document.all.t.rows(2).cells(0).innerHTML = "";
        alert( "文件内容为空,请重新读取" );
        return;
      }
    else  {
          var lineNum = 0;
          var str     = "";
          //读第一个字符,如果曾加过编号,该字符一定为"<"
          var hasNumed = textFile.read(1);
          while ( !textFile.AtEndOfStream )
          {
              lineNum ++;
              if (hasNumed != "<")
                  str = str + "<b>" + lineNum + ":</b>" ;
              if (lineNum == 1) str += hasNumed;
              while ( ! textFile.AtEndOfLine )
              {
              ch = textFile.read(1);
              str += ch;
              }
        str += "<br>";
        if( ! textFile.AtEndOfStream ) textFile.skipLine();
        }
      document.all.t.rows(2).cells(0).innerHTML = str;
     }
     textFile.close();
     //显示下面隐藏的表格(带有批语的文本框)
     document.all.t1.style.visibility = "visible";
     document.all.annotation.focus();
   }
  //////////////////////////////////////////////////////////////////////////
  // 调用批注窗口,打开一个网页模式对话框,新建一个批注
  // 传入参数, para[0] -- 选择的文本串
  //           para[1] -- 批注的内容,新建批注时,该参数值为空
  // 返回参数, val[0] -- 批注窗口操作(new,modi,dele,cancel),
  //           val[1] -- 批注内容
  function contProc()
  {
```

```
    var str = document.selection.createRange().text;
    if  (isEmpty(str))
         return false;
    var para = new Array();
    para[0] = str;     // text selected
    para[1] = "";      //批注初始值
    var sfe = "dialogWidth:25;dialogHeight:20";
    var val = window.showModalDialog("desCreator.htm",para,sfe);
    switch (val[0])
    {
      case "new":
         var description = val[1];
         if (description != null && !isEmpty(description ))
             updateHtmlFile(str,description );
         break;
      case "modi":
          alert("modi");
          break;
      case "cancel":
         break;
      }
   return false;
  }
//////////////////////////////////////////////////////////////////////////////////
// 新加批注后,如果选择保存,则修改 HTML 文件
// 注意:两次批注不能选中完全相同的文本
  function updateHtmlFile(strSelected,description)
  {
    var desFile, fileContents;
    // get the contents of htm file
   if (fileObj.FileExists(savePath + saveName))
   {
     desFile = fileObj.OpenTextFile(savePath+saveName,forReading);
     if (desFile.AtEndOfStream)
         fileContents = document.all.t.rows(2).cells(0).innerText;
     else
         fileContents = desFile.readAll();
     desFile.close();
     } else fileContents = document.all.t.rows(2).cells(0).innerHTML;
     //id of div with description
     var newDivID = "d_" + getDivNum(fileContents,"div");
     var linkName = "A_" + newDivID.substring(2);
     var strWithDescription = "<a href=# name='" + linkName + "'"
            + "onmouseover=showDes(1,'" + newDivID + "') "
            + "onmouseout=showDes(0,'" + newDivID + "') "
            + "onclick=\"" + "desModify(" + "'" + strSelected + "','" + linkName + "')\"" + ">"
            + strSelected + "</a>";
     if (!isEmpty(description))
         fileContents = fileContents.replace(strSelected,strWithDescription);
     //(1)更新文件内容
     var newFile = fileObj.CreateTextFile(savePath+saveName);
```

```
        newFile.write(fileContents);
        //(2)将批注内容定义到一个图层中,隐藏显示
        var desStr = "<table bgcolor = #00ccff>" +
                     "<tr><td>" + description + "</td></tr>" +
                 "</table>";
        var mystyle = "visibility:hidden;position:absolute;left:" + window.event.clientX
                 + ";top:" + window.event.clientY;
      //alert(mystyle);
      newFile.write("<div id = " + newDivID + " style = " + mystyle + ">" + desStr + "</div>");
      //(3)将一段脚本程序写到文件中
      var scriptStr = "<script language = javascript>" +
                       "function showDes(bz,ob){" +
                       "v = \"visible\"\;" +
                       "if(bz==0) v = \"hidden\"\;" +
                       "document.all(ob).style.visibility = v;" +
                       "}" + "</" + "script>";
      scriptStr + = "<link href = \"mypub.css\" rel = \"stylesheet\" type = \"text/css\">";
      if ( fileContents.indexOf(scriptStr,0) == -1 )
         newFile.write(scriptStr);
      newFile.close();
      //read content to cells again so that link can be displayed
      readFile();
}
////////////////////////////////////////////////////////////////////////////////////
// 求下一个图层的编号
function getDivNum(fileCon,ob)
{   //create div's id automatically
     var temp      = fileCon;
     var obStr     = "<" + ob;
     var count     = 0;
     while (temp.indexOf(obStr,0) ! = -1)
     {
         count++;
       temp = temp.substring(temp.indexOf(obStr,0) + 1);
      }
     count++;
     return count;
 }
////////////////////////////////////////////////////////////////////////////////////
// 用于显示图层,该函数一并被写入到创建的.htm 文件尾部
function showDes(bz,ob)
{
   v = "visible"
   if (bz = = 0) v = "hidden";
   document.all(ob).style.visibility = v;
}
////////////////////////////////////////////////////////////////////////////////////
// 根据 linkName,获取批注文本
function getDescription(linkNameStr,desFileCon)
{
     var divName = "d_" + linkNameStr.substring(2);
```

```
        var divStr = desFileCon.substring( desFileCon.indexOf( "id=" + divName ) );
        // 注意<TD>标记要大写
        var description = divStr.substring( divStr.indexOf ("<TD>") + 4, divStr.indexOf
("</TD>"));
        return description;
 }
////////////////////////////////////////////////////////////////////////////////
// 单击批注处,修改批注内容或删除批注
// 返回值 val[0]存储对批注所作的操作--修改(modi)、删除(dele)
function desModify(strSelected,linkName)
{
   var desFileCon = document.all.t.rows(2).cells(0).innerHTML;
   var description = getDescription(linkName,desFileCon);
   var para = new Array();
   para[0] = strSelected;     //text selected
   para[1] = description;     //批注内容
   var val = window.showModalDialog ("desCreator.htm",para,
                                  "dialogWidth:25;dialogHeight:20");
   switch (val[0])
   {
          case "dele":
               deleteDes(strSelected,linkName);
               break;
          case "modi":
               var tempFlie;
                tempFile = fileObj.OpenTextFile(savePath + saveName,forWriting);
               // 将文档中的旧批注用新的替换
               newstr = desFileCon.replace(getDescription(linkName,desFileCon),val[1]);
               tempFile.write(newstr);
               tempFile.close();
               break;
          case "cancel":
               break;
     }
     readFile();
     return false;
}
////////////////////////////////////////////////////////////////////////////////
// 删除批注所建立的连接信息和图层,首先求出对应的字符串,然后用空替换
//
function deleteDes(strSelected,linkName)
{
    var linkText = document.all(linkName).innerText;
    var linkStr = document.all(linkName).outerHTML;
    //(1)delete link part
    var divID = "d_" + linkName.substring(2);
    var str2 = fileObj.OpenTextFile(savePath + saveName,forReading).readAll();
    var linkWithDesStr = "<a href=# name='" + linkName + "'"
                 + "onmouseover=showDes(1,'" + divID + "') "
                 + "onmouseout=showDes(0,'" + divID + "') "
                 + "onclick=\"" + "desModify(" + "'" + strSelected + "','" + linkName + "')\"" + ">"
```

```
                + strSelected + "</a>";
        str2 = str2.replace( linkWithDesStr,linkText );
        //(2)delete div part
        var pos1 = str2.indexOf("<div id = d_" + linkName.substring(2),0);
        var pos2 = str2.indexOf("</div>",pos1);
        var divStr = str2.substring(pos1,pos2 + 6);
        str2 = str2.replace(divStr,"");
        var newFile1 = fileObj.OpenTextFile(savePath + saveName,forWriting);
        newFile1.write( str2 );
        newFile1.close();
      }
    </script>
    </head>
    <body bgcolor = "#CCCCCC">
    <table id = t width = "600" border = 1 bordercolorlight = "#333333" bordercolordark = "#ffffff">
      <tr height = 30>
          <td align = center style = "font - size: 18; font - weight: bold" >文本文档批注系统</td>
      </tr>
      <tr height = 20>
          <td>
            选择文件<input type = "file" name = "homework" onchange = "readFile()"
                          title = "提示: &#13;选择要批注的文本文件">
         </td>
      </tr>
      <tr><td oncontextmenu = "javascript:return contProc();"></td></tr>
      <tr>
      <td>
         <table id = t1 width = "587" border = "0" style = "textalign: center; visibility: hidden">
           <tr>
              <td align = center width = "15">批<br><br>语</td>
              <td width = "558">
                  <textarea name = annotation rows = 7 cols = "90"></textarea><br>
                  <input type = "button" value = "保存批语" onclick = "saveAnnotation()">
              </td>
            </tr>
         </table>
         </td>
      </tr>
    </table>
    </body>
```

上述代码完成后，我们验证一下运行情况，第一次对某个文本文件进行批注时显示原始文件，如果已经批注过，第二次打开时，将显示批注的文本，界面如图 5-18 所示。

**几点说明：**

① 对于<input type="file"…>输入控件，其功能是选择一个文件。在本例中，对于选择的文件，我们需要打开并显示。因此，我们设计通过 onclick = " readFile ( ) " 来激活 readFile 读取文件并显示，验证一下，并不能达到我们的目的，当单击<input type="file"…>输入窗口后面的“浏览…”按钮后，只是打开文件选择窗口，选择文件后，文件并没有显示。为什么呢？




图 5-18 选择文件并显示

文件没有显示，首先分析可能是单击鼠标时 readFile()没有被执行，因此，我们再设置一个事件为 onchange="readFile()"，即：

```
<input type="file" name="homework" onclick="readFile()" onchange="readFile()" title="… ">
```

再次运行，运行情况达到了预期的效果，单击“浏览…”按钮，选择文件后，马上显示。上述代码虽然结果正确，但还是有问题的，因为 readFile()不可能执行两次。因此，这就说明通过 onclick 事件来激活 readFile()可能是有问题的，为此，我们将代码写成 onclick="alert('hi')"。再来验证，结果发现当我们单击“浏览…”按钮时，首先打开了 alert 窗口，然后才打开文件选择窗口。这样问题就很清晰了，如果设置 onclick="readFile()"，则 readFile()先执行，而此时还没有选择文件，当然就不显示了。

根据上述分析，应使用 onchange 来激活 readFile，正确的写法是：

```
<input type="file" name="homework" onchange="readFile()" title="… ">
```

再次来验证，一切都正确了。通过这个例子，也可以看出对于事件，必须要清楚其激活顺序，否则代码的执行可能达不到预期的效果，这也是面向对象编程的难点。

② 在 updateHtmlFile(strSelected,description)中，用到了大量的字符串操作，稍有疏忽，将导致错误，并且因为 JavaScript 脚本语言没有好的调试工具，错误查找很难。最容易引起错误的是英文单引号(')和双引号(")的应用。

例如，在<a>标记中，对于 onclick 属性的设置，属性值两边的双引号(")不能省略，应确保生成的代码形式为：onclick="desModify(' ',' ')"。否则，如果选择批注文本中包含空格时，单击则不起作用，不包含空格时不存在问题。

对于上述情况，我们换用 onclick="alert('hello')"进行试验，发现没有问题。但是一旦换成 onclick=desModify(' ',' ')，结果就不对了，即使函数内部只包含一个 alert。理论上

讲，在 HTML 中，属性值两边可以不写""，错误的调试非常艰难。用 FrontPage 打开创建的 HTML 文件，如果选择的文本中含有空格时，发现生成的代码在 FrontPage 中的解析错误，遇到第一个空格后，就认为 onclick 属性设置结束。这显然是不正确的，如果把属性值 desModify('','') 两边加上""，即为：onclick="desModify('','')"，在 FrontPage 中马上就看到了正确的解析结果。这可能是 JavaScript 和浏览器本身存在的一个 bug 吧。

③ 在一个 Table 中，如果单元格中包含多行文本框(<textarea >)，它的属性 cols 的值直接影响表格和单元格的 width 属性，如果设置得太大，则体现为即使修改表格和单元格的 width 属性大小，表格似乎也无法缩小。实际上这是<textarea >标记的 cols 设置太大造成的，是它把表格撑大了。

④ oncontextmenu 事件是 IE 新版本中支持的事件，当右击鼠标时发生该事件，可以用于<body>标记和许多其他的标记，如果要禁止在某个元素上右击鼠标，可以赋值 false，例如，<body oncontextmenu="alert('鼠标右键已禁用')；self.event.returnValue=false">，则用户在网页上右击鼠标时弹出一个警告窗口，而不是原先的快捷菜单。

⑤ 网页调用问题，除了传统的通过超链接来打开一个网页外，上述代码使用了 window.showModalDialog()函数，可以打开一个新的网页，并进行参数传递。在本例中，当用户选择了一段要批注的文本后，在选择文本上右击鼠标，则调用 contProc()，打开批注处理页面文档 desCreator.htm，在该文档中输入新的批注、更改已有批注内容或删除批注。

⑥ 在某些网页中，不能使用鼠标拖动的方法或通过右击鼠标的菜单选择选中文字，通常是因为网页中设置了<body>的相关属性，此时，可以通过设置浏览器的相关属性来解决。具体方法是：选择"工具"菜单中的"Internet 选项"，在打开的"Internet 选项"对话框中选择"安全"选项卡，单击"自定义级别"按钮，打开"安全设置"对话框；在"安全设置"对话框中，将"活动脚本"和"java"设置为"禁用"；然后按 F5 键刷新网页。此时，网页中的内容就可以复制了。需要说明的是，为了保证浏览器的正常工作，复制完成后，应恢复浏览器的默认设置。

### 3. 批注、批注修改、删除批注功能设计

用户可以选择一段文本，然后在选中的文本上右击鼠标，即可对选择的文本添加批注。当用户右击鼠标时，将打开一个新的窗口文档 desCreator.htm。这是进行批注的添加、修改和删除的页面。批注处理页面 desCreator.htm 设计如图 5-19 所示。




图 5-19　批注的添加、修改和删除页面

单击“保存”按钮，返回，可以看到第七行称为一个超链接，如图 5-20 所示。

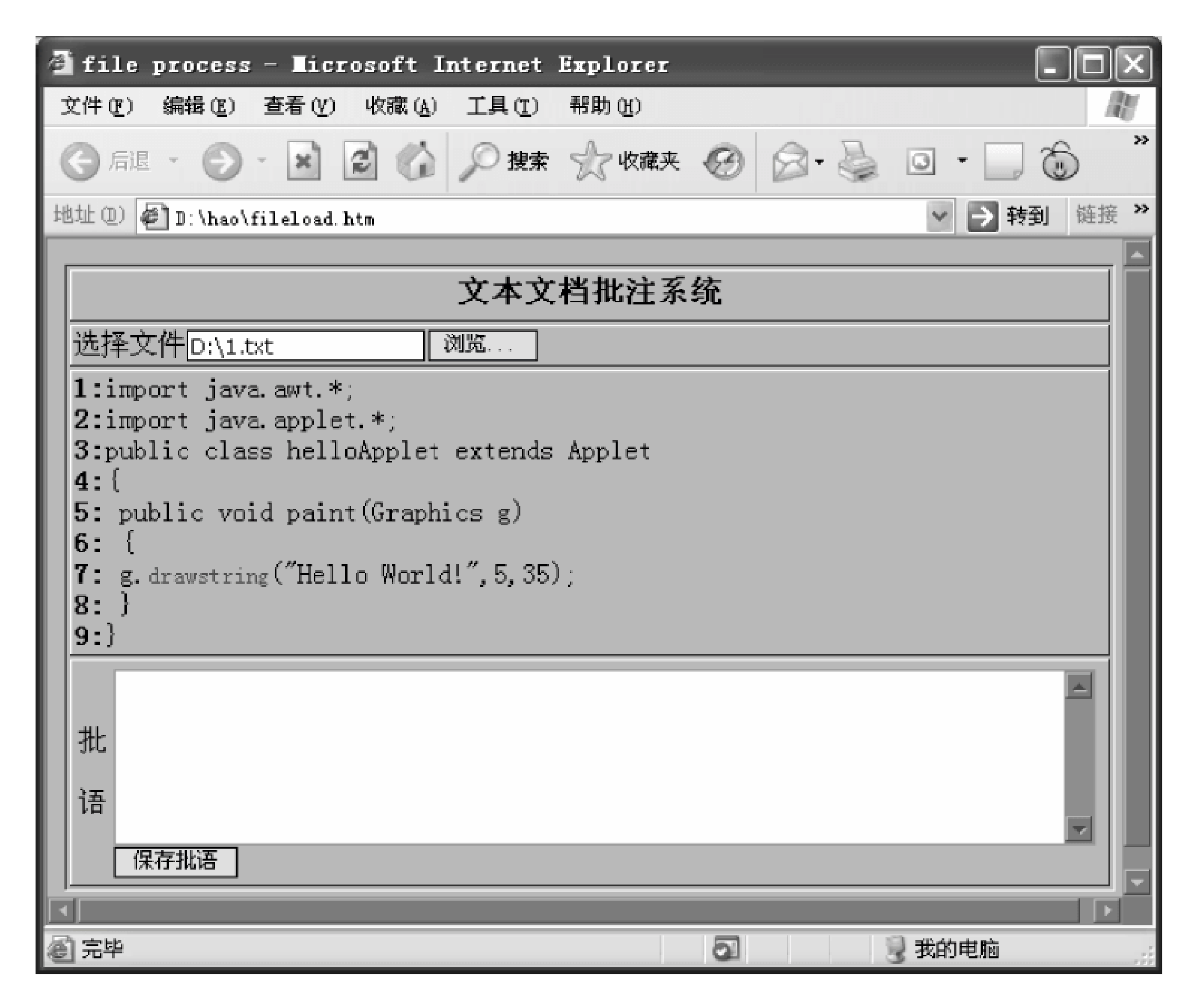



图 5-20 增加批注后的文档

当鼠标移向批注文本时，显示批注内容，单击批注文本，则再次打开批注处理页面，可进行批注的修改或删除。

desCreator.htm 文档内容如下：

```
<html>
<head>
<title>注释的添加、修改和删除</title>
<script language = "javascript" src = "myglobal.js"></script>
<script language = "javascript">
/////////////////////////////////////////////////////////////////////////////
// 在窗口中显示选择的文本和批注内容，以便于修改批注或删除批注
// recValue[0] -- 选择的文本
// recValue[1] -- 已有批注
  var recValue = window.dialogArguments;
  function dispInitDes()
  {
      var strSelected = recValue[0];
      var description = recValue[1];
      document.all.t.rows(1).cells(0).innerText = strSelected;
      document.all.destext.value = description;
}
/////////////////////////////////////////////////////////////////////////////
// 关闭窗口时调用，修改或删除批注，设置网页对话框返回值，存储在 retValue 数组中
// retValue[0]-- 批注窗口操作(modi,dele,cancel)
// retValue[1]-- 新的批注内容
```

```
// 具体的更新文件在 fileload.htm 中完成
function editDes()
{
        var retValue = new Array();
        description = document.all.destext.value;
        // 新建或修改批注
        if (description ! = "")
        {
          if (recValue[1] =="" )
               retValue[0] = "new";
          else
               retValue[0] = "modi";
          retValue[1] = document.all.destext.value;
             confirmOption( "确定要保存吗?",retValue );
        }
        else // 批注区为空
        {
             if (recValue[1] =="" )
             {
                  alert( "注释内容不能为空!" );
                  document.all.destext.value = "";
                  document.all.destext.focus();
                  return;
             }
             else {
                  retValue[0] = "dele";
                  confirmOption( "确定删除此批注吗?",retValue );
                  return;
            }
       }
}
function confirmOption(prompt,value)
{
   var ok = confirm(prompt);
   if( ok ) {
             window.returnValue = value;
             window.close();
    }
    else {
          document.all.destext.focus();
          return;
    }
 }
 ///////////////////////////////////////////////////////////////////////
 // 取消批注的编辑,返回一个 cancel
 function cancelWin()
 {
    var retValue = new Array();
    retValue[0] = "cancel";
    retValue[1] = recValue[1];
    confirmOption( "直接关闭将不会保存填写的批注,确认要关闭吗?",retValue);
```

```
}
////////////////////////////////////////////////////////////////////////////////
// 键盘操作,F2--确定,Esc--取消
function doByKey()
{
  if ( window.event.keyCode==113) //F2
        editDes();
  if ( window.event.keyCode==27) //Esc
        cancelWin();
}
</script>
</head>
<body bgcolor = "#CCCCCC" onload = "dispInitDes()" onkeydown = "doByKey()">
<div align = "center">
  <center>
  <table border = "0" cellpadding = "0" width = "100%" height = "100%">
    <tr>
        <td width = "100%">
        <table id = t border = 1 width = 300 bordercolorlight = #333333 bordercolordark = #ffffff
align = center >
          <tr>
            <td height = 50 align = center style = "font-size: 18; font-weight: bold">批注
输入、修改与删除</td>
        </tr>
        <tr> <td> </td> </tr>
         <tr>
         <td>
             请在下面书写批注:<br>
             <textarea id = destext cols = 50 rows = 6></textarea><br><br>
             <input type = button value = 确定(F2) onclick = "editDes()"> 
             <input type = button value = 取消(Esc) onclick = "cancelWin()">
         </td>
         </tr>
       </table>
       </td>
       </tr>
    </table>
    </center>
</div>
</body>
```

新建批注后,当用户将鼠标指向包含批注的文本时,下方显示批注内容,如图 5-21 所示。

文本文档批注系统允许用户添加、修改和删除批注,同时还允许用户添加一个总体的评价意见或说明。

#### 4. 批注生成的 HTML 文档

新建批注后,在 C 盘根目录下创建一个同名的文件夹,包含创建的同名.htm 文件。在原先文档的基础上生成一个同名的 HTML 文档,被批注的文本被替换为一个带有超链接




图 5-21　显示批注后的文本

的书签定义。批注内容以<div>块的形式保存在系统创建的 HTML 文档中。此外，该文件还包含用于显示图层的 showDes()函数，以方便该文件单独在浏览器中打开。

批注完成后，用浏览器打开生成的批注文件，显示界面如图 5-22 所示。

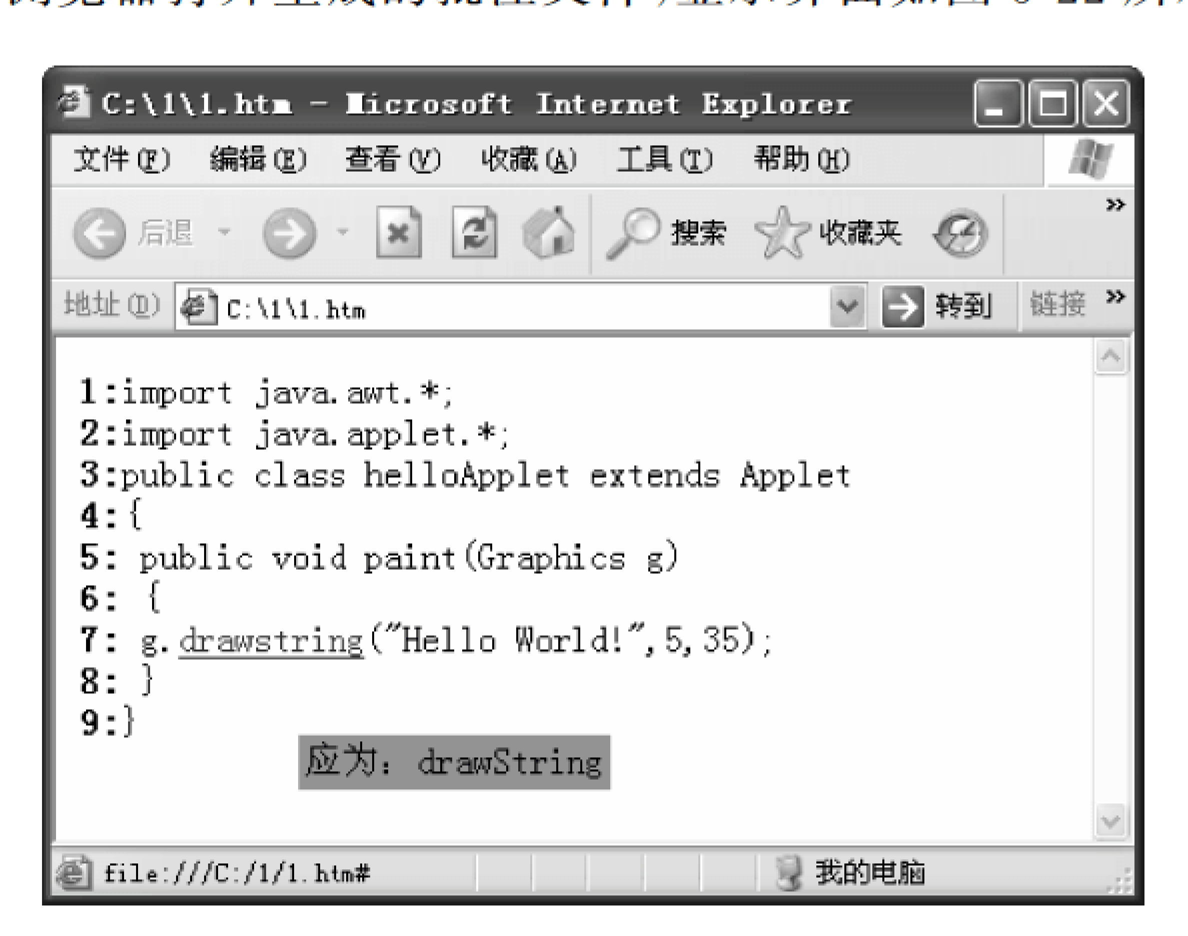



图 5-22　创建的同名.htm 文件

在浏览器中，执行“查看”菜单中的“源文件”可以查看批注生成的 HTML 代码，可以更好地理解上述的程序。通过这些生成的页面源代码，可以查找程序中遇到的错误，特别是 string 的查找、替换操作，大小写的误写可能直接导致程序的运行错误，如将<td>写为<TD>等，甚至标记中的属性值的两侧是否有双引号""等。

#### 5. 小节

文本文档批注系统比较全面地解释了 JavaScript 中关于文件的处理问题，用到了 window 对象的一个新方法 window. showModalDialog(网页模式对话框)，它和第一个 E-learning 系统相比对于参数的传递用了不同的方法，使得在 Web 编程中可以方便地实现传统程序设计中对话框的功能。

进行 JavaScript 的调试是件很复杂的事情，因为缺少有效的调试工具，在调试过程中，一些小的错误可能会花费很长的调试时间，特别要注意字符串的操作，牵扯到单引号、双引号等一系列问题，还有就是字符串中字母的大小写问题。

在程序调试中，在 fileload. htm 中的 getDivNum()中容易出现问题，因为我们是按照小写的<div>来查找现有的批注个数，来生成新的批注编号。用记事本打开生成的 HTML 文件，我们发现，添加的<div>等标记，有时候是小写，有时候则变成大写，这样函数 getDivNum()计算的批注编号将是错误的。我们不能解释导致这种情况发生的原因，可能是系统存在的 bug。对于这种情况，要增加程序的鲁棒性，可以在生成的新文件中的开始处或结尾处增加一个记录当前批注数量的数字，这个数字在新建批注或删除批注时需同时进行修改。

### 5.10.3 创建折叠式菜单

在许多网页上，都有折叠式菜单，这类菜单的创建可以利用 HTML DOM，通过纯 JavaScript 程序编码实现。目前常用的菜单有树形目录结构的，也有其他形式的折叠式菜单，下面代码演示了一个简单的折叠式菜单创建过程，文件名为 mymenu. htm。

折叠式菜单网页 mymenu. htm 代码清单如下：

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>折叠式菜单</title>
<style type="text/css">
 A:hover { color:#FF0000;text-decoration:none}            <!-- 指针移到超链接上的样式 -->
 A{color:#0000FF;text-decoration:none}                    <!-- 超链接的样式 -->
.menutitle{cursor:hand;margin-bottom: 5px;background-color:#C48700;
   color:#000000;width:200px;padding:2px;text-align:left;
   font-size:13px;border:1px solid #880088}               <!-- 标题菜单项样式 -->
.submenu{margin-bottom: 5px;background-color:#CCE34A;line-height:180%;
   color:#000000;width:200px;padding:2px;text-align:left;
   font-size:13px;border:1px solid #000088;display:none}<!-- 子菜单项样式 -->
</style>
<script type="text/javascript">
 ////////////////////////////////////////////////////////////////////////////////////
 // 创建菜单的函数
 //
  function createMenu()
  {
```

```
  var node = document.getElementById("masterdiv");        //获取 div 元素
  for(i = 0;i<menuitems.length;i++)                       //对所有的菜单项循环
  {
    id1 = menu_id[i];
    text1 = menu_text[i];
    cha0 = id1.substring(0,2);                            //获取章代码
    cha1 = parseInt(cha0);
    sec1 = parseInt(id1.substring(2,4));                  //获取节序号
    // 一级菜单处理
    if (sec1==0)   //节序号为 0 时为章标题菜单,即该菜单项包含子菜单
    {
         var cha_node = document.createElement("span");  //建立章标题元素
         cha_node.destination = "sub" + cha0;    //绑定该章标题下的节子菜单元素的 id 标识
         cha_node.onclick = function()
         {
             switchMenu(this.destination);
         };
         cha_node.className = "menutitle";                //指定元素的样式类
         cha_node.innerHTML = text1;                      //设置章标题文本
         node.appendChild(cha_node);                      //添加为 div 元素的子元素
         var br_node = document.createElement("<br>"); //建立“换行”元素
         node.appendChild(br_node);                       //在 div 下添加“换行”子元素
    }
    // 二级菜单处理
    else
    {
      if (sec1==1)                                        //二级菜单中的第一个菜单项
      {   var sec_node = document.createElement("span"); //建立节标题子菜单元素
          sec_node.id = "sub" + cha0;                     //设置节标题子菜单元素的 id 属性
          sec_node.className = "submenu";                 //指定样式类
          sec_node.innerHTML = "<a href = 'p" + id1 + ".htm'>" + text1 + "</a>";
                                                          //设置内部 HTML 文本
          node.appendChild(sec_node);    //将元素添加为 div 元素的子元素
      }
      else //二级菜单中的其他菜单项
      {
          var sec_node = document.getElementById("sub" + cha0);    //得到它所在的子菜单
          ccc = sec_node.innerHTML;    //内部 HTML 文本中增加一个新行
          sec_node.innerHTML = ccc + "<br><a href = 'p" + id1 + ".htm'>" + text1 + "</a>";
      }
    }
  }
}
//////////////////////////////////////////////////////////////////////////////////////
// 此函数用来打开和关闭节标题子菜单
//
function switchMenu(obj)
{
      if(document.getElementById)
      {
          var el  =  document.getElementById(obj); //获取子菜单元素,下一行为获取 span 元
```

```
                                                        素数组
            var ar = document.getElementById("masterdiv").getElementsByTagName("span");
            if(el.style.display ! = "block")        //当前的节标题子菜单处于关闭状态时
            {
                for (var i = 0; i<ar.length; i++)   //关闭已经打开的节标题子菜单
                {
                  if (ar[i].className=="submenu")
                      ar[i].style.display = "none";
                }
            el.style.display = "block";             //打开当前有节标题子菜单
           }
           else
           {  el.style.display = "none"; } //打开当前有节标题子菜单,已经打开时将它关闭
      }
   }
   ////////////////////////////////////////////////////////////////////////
   // 菜单项数据部分,菜单代码约定如下:每个菜单代码占四位,前两位对应一级菜单,
   // 后两位为二级菜单。对于一级菜单,菜单项代码后两位为 00
   //
   content = "0100#第 1 章 客户端开发";
   content + = "&0101#1.1 浏览器技术";
   content + = "&0102#1.2 客户端脚本语言";
   content + = "&0200#第 2 章 服务器端开发";
   content + = "&0201#2.1 Web 服务器";
   content + = "&0202#2.2 服务端脚本程序";
   content + = "&0203#2.3 数据库技术";
   var menu_id = Array(30);             //数组用来保存所有菜单项的代码
   var menu_text = Array(30);           //保存所有菜单项的文本
   var menuitems = content.split("&");  //将数据以"&"为分隔符分开,形成数组 menuitems
   for(i = 0;i<menuitems.length;i++)
   {
     var oneitem = menuitems[i].split("#");
     menu_id[i] = oneitem[0];           //保存菜单项代码,形成菜单项对应的超链接 URL
     menu_text[i] = oneitem[1];         //保存菜单项文本
   }
</script>
<base target = "main">
</head>
<body topmargin = "20" leftmargin = "10" onload = "createMenu()">
<p>利用 JavaScript 创建折叠式菜单示例</p>
<!-- 菜单所在的 div 标记 -->
<div id = "masterdiv" align = "left">
</div>
</body>
</html>
```

在浏览器中打开上述网页,通过设置每一个菜单项对应的超链接,可以很好地实现网页之间的控制。创建的折叠菜单如图 5-23 所示。

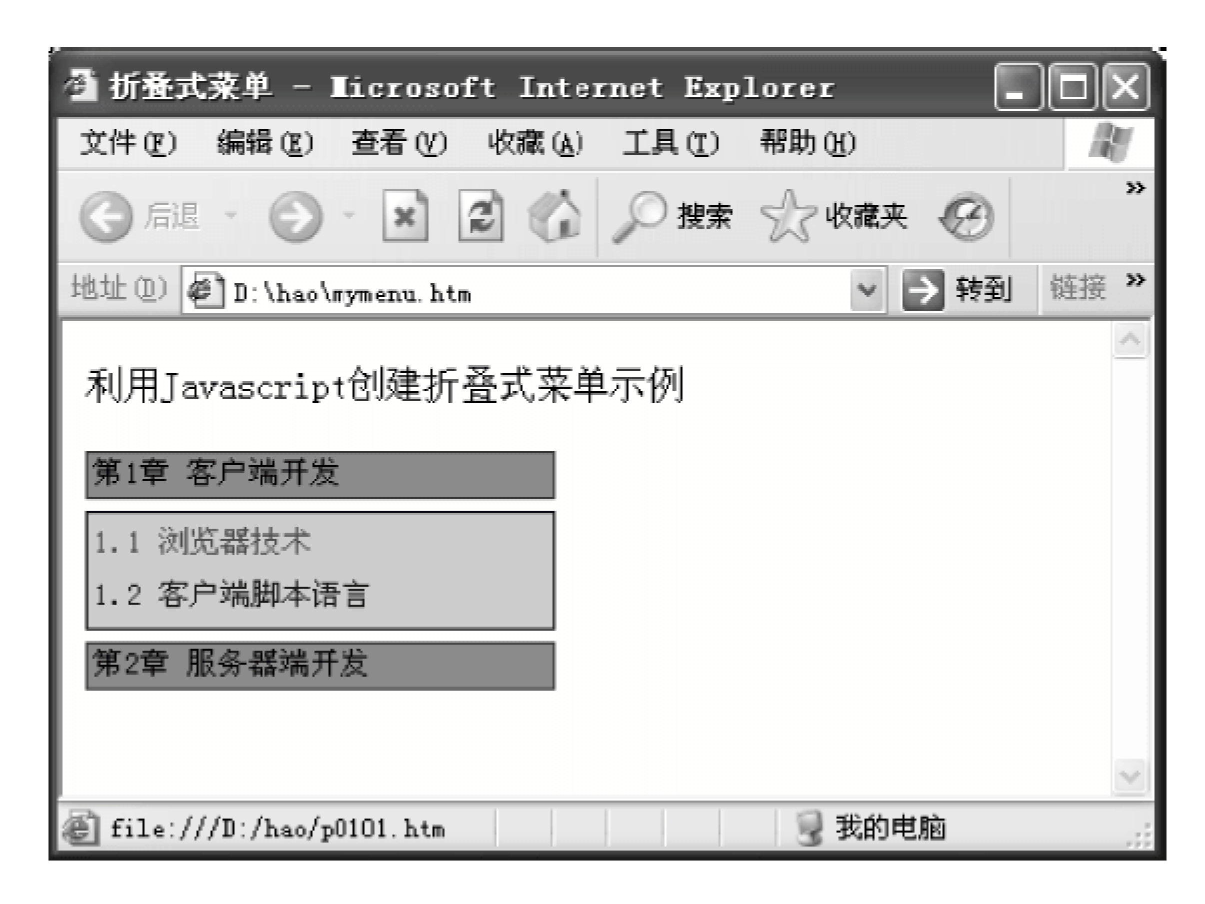



图 5-23　折叠式菜单示例

当用户在一级菜单上单击时，将打开对应的二级菜单。二级菜单对应具体的超链接，当用户鼠标指向二级菜单项目时，在浏览器状态栏，可以看到对应的超链接。用户可以根据需要，修改每一个二级菜单项目对应的超链接，就可以很好地实现页面之间的控制。

在第 4.6 节，我们也看到了上述折叠式菜单的创建方法。相比而言，这里的方法更复杂。我们给出上述方法的原因有两个：(1)学习 HTML DOM 的操作方法；(2)在上述代码中，我们没有手工地将菜单值写在<body></body>内，而是通过脚本程序定义在了一个数组内，然后通过 CreateMenu()函数来生成 html 文档树即在 masterdiv 下增加节点，这样的思路可以实现很多问题的形式化，从而实现自动化。

# 思　考　题

1. 如何理解浏览器和客户端脚本引擎之间的关系？

2. JavaScript 语言有哪几个组成部分？简述各个部分的功能。

3. 在 JavaScript 中，myArray = new Array(10)是什么意思？如何定义一个 3×4 的二维数组？

4. 什么是网页对象？简述 window 对象和 document 对象常用的属性和方法。

5. 画出 HTML DOM 的对象层次图。文档对象 document 有哪些常用的属性和方法？

6. 什么是 IntelliSense 技术？如何知道一个 HTML 标记或一个 JavaScript 内部对象有哪些属性或方法？

7. 有哪些常用的浏览器事件？它们是如何触发的？

8. 如何设置 body 对象，以禁用页面上的右键来防止复制页面内容？

9. 结合图像地图的实例和 5.10.2 节的“文本文档批注”实例，编写一个地图系统，要求当鼠标指向某个省或直辖市时，在鼠标右下角打开一个多行矩形区域，滚动显示该热点的相关信息。

10. 根据 5.10.1 和 5.10.2 小节的综合案例，总结一下两个 htm 页面之间传递参数的方法。

11. 若在页面 z.htm 中调用页面 l.htm 中的一个函数，如何调用？

# 第 6 章

# 服务端开发

对于 Web 应用,大量的数据管理工作和业务逻辑都是通过服务端程序实现的。服务器编程,与 Web 服务关系密切。针对不同的 Web 服务器环境,所用的开发工具也不相同。目前,常用的 Web 服务端开发环境包括 ASP、JSP 和 PHP,它们分别适用于不同的 Web 服务器。例如,Windows 平台中的 Internet 信息服务 IIS,支持 ASP 开发。如果要使用 JSP 开发 Web 应用,则需要在 Web 服务器上安装 Tomcat 应用服务器。

在 Web 应用的开发中,Java 技术以其平台无关性受到了开发人员的欢迎,作为 Java 技术的一种实现,结合 Servlet 和 JavaBean,使得 JSP 成为众多 Web 应用首选的开发工具。本章将围绕 Java 技术,详细介绍 Web 应用中服务器端的开发问题。

## 6.1 B/S 三层体系结构与 Web 服务器脚本程序

服务端脚本程序是指在 Web 服务器上运行的程序。在 B/S 模式下,当用户下载一个网页时,如果网页中包含服务端脚本程序,Web 服务器将首先执行网页中的脚本程序,然后把执行的结果网页发送到客户端浏览器显示。

### 6.1.1 B/S 三层体系结构

在 Internet 环境,客户机/服务器(C/S)两层体系结构表现出了许多局限性,它不能跨越企业局域网,缺少可伸缩性,客户端维护复杂等。为此,出现了基于 Web 的浏览器/服务器(B/S)三层体系结构(3-tier)。这种体系结构大致可以定义为:客户机层上的表示层、中间的 Web 服务器层和后端的数据库服务器层。B/S 三层体系结构如图 6-1 所示。

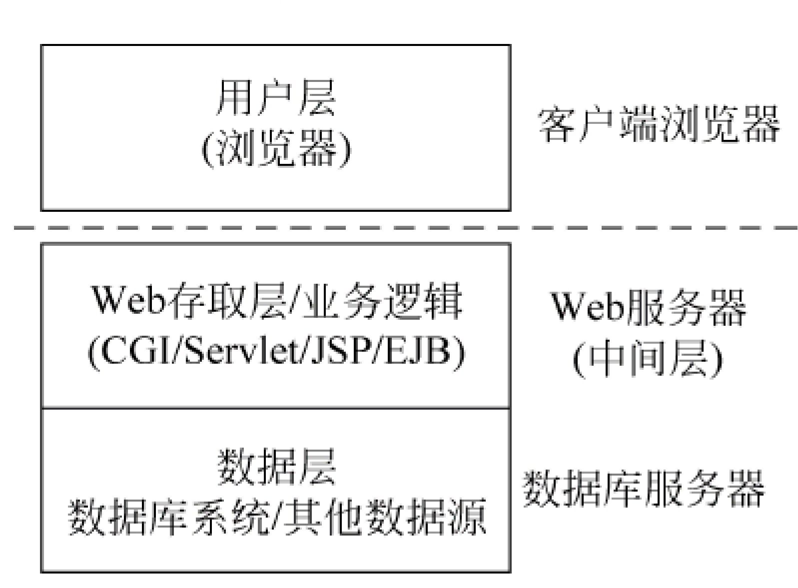



图 6-1 B/S 三层结构的分层功能界定

在 B/S 三层体系结构模式下,客户端不再需要安装特定的客户端应用程序,取而代之的是通用浏览器软件,所有的用户业务逻辑都被部署在新的中间层上。新的中间层往往是一组公共网关接口(CGI)程序,CGI 程序常常充当一种中间软件,从 Web 浏览器接收请求,决定必须调用哪些计算资源

来满足这些请求，并向浏览器发回响应。传统的 CGI 程序一般是由 C 或 C++开发的，这些语言的执行速度较快。随着 JavaServlet 的出现，同样作为连接前端 Web 请求与后端数据资源的中间层组件，JavaServlet 可以以更高的效率和可移植性来实现 CGI 的功能，并最终会取代传统的 CGI 程序。

由于三层体系结构通常是基于 Web 的，所以中间层应用程序通常工作在 Web 服务器上，被视为 Web 服务器的一种功能扩展，因此中间层又称为 Web 服务器层。在 Web 服务器上，通过大量的包含 CGI/Servlet 的服务器脚本程序页面，接受来自客户端浏览器的请求，以及完成对数据库的操作。

### 6.1.2　脚本引擎与服务端脚本程序

什么是脚本引擎呢？简单地讲，脚本引擎就是指脚本程序的运行环境，负责脚本程序的解释，来具体处理用相应脚本语言书写的脚本命令。例如，ASP 脚本语言必须运行在 IIS（Internet Information Server）上；Tomcat 是 JSP 和 Servlet 的容器，运行 JSP 网页必须安装和配置 Tomcat。没有脚本引擎，脚本程序就不能运行。

在 ASP 结构中，ASP 解释器（ASP.DLL）负责 ASP 页内服务器端脚本程序的解析任务。这需要安装相应脚本程序语言的脚本引擎，即脚本程序解释器，来具体处理用相应语言书写的脚本命令，它们以 COM 组件的形式供 ASP 解释器调用。Active Server Pages 带有两个脚本引擎：Microsoft Visual Basic Scripting Edition（VBScript）和 Microsoft JScript，其中 ASP 中的默认语言是 VBScript。在 Windows 平台上，Internet 信息服务器 IIS 内含 Active Server Pages，安装 IIS 后，VBScript 和 JScript 脚本引擎会自动地安装在服务器上。

如果要使用其他的脚本程序语言，例如 JSP，则必须在 Web 服务器上安装相应的脚本引擎，即 Tomcat 应用服务器。脚本引擎将遵循 ActiveX 脚本标准并作为一个 COM（组件对象模型）对象驻留在 Web 服务器上。

在 Web 服务器上安装了服务端脚本引擎后，进行简单的配置，然后就可以在网页中编写服务端程序了。和客户端脚本程序书写在<script></script>标记对内不同，服务端脚本程序一般书写在定界符“<%”和“%>”内。用户可以在网页文件的开头，通过<%@ language %>指令来设置服务端脚本程序语言，一般形式是<%@ language=ScriptingLanguage %>。

例如，下面是一个包含服务端脚本程序的网页（test.jsp），代码清单如下：

```
<%@ page contentType="text/html;charset=gb2312" %>
<html>
<head>
<title>Hello,JSP</title>
</head>
<body>
  <% out.println("你好, JSP!"); %> <br>
    现在的时间是：
  <%= new java.util.Date() %>
</body>
</html>
```

通过浏览器下载该页面时,Web 服务器将执行其中的服务端脚本程序,在客户端浏览器中显示执行后的结果。

## 6.2 Java 程序设计

Java 技术是 Sun Microsystems 于 1995 年推出的一种极富创造力的计算平台。最初称为 OAK,1995 年被重命名为 Java 编程语言。Java 技术为用户带来了无数令人兴奋的可能性,它几乎使所有应用程序都能在任何计算机或设备上运行。Java 技术的多功能性、有效性、平台无关性以及安全性已经使它成为网络计算领域最完美的技术,给未来的计算模式产生了革命性的影响,是继 HTML 后,Internet 发展的又一个里程碑。今天,Java 技术已经无处不在,从桌面 PC 到科学超级计算机和互联网,从移动电话到移动手持设备,从家庭游戏机到信用卡,几乎在所有的网络和设备上都会看到 Java 技术的身影。

### 6.2.1 Java 语言的特点

Java 程序设计语言是 Sun 于 20 世纪 90 年代初开始研发的,总设计师为 James Gosling。对于习惯了 C++的程序员来说,Java 有许多独到的特点:

(1) Java 没有主函数和全局函数

C++并非完全意义上的面向对象语言,最明显的例子是,在 C++中,必须有一个独立的主函数(在 DOS 和 UNIX 下是 main(),在 Windows 下是 WinMain()),还可以定义可以直接使用的大量的全局函数或者使用 C++中的命名空间“extern C”来使用原有的 C 过程调用。

Java 是完全面向对象的语言,它没有主函数等完全孤立的东西,任何函数都必须隶属于一个类。当然,任何程序都有一个入口,Java 程序也有主入口函数,名称同样是 main(),但它必须包含在一个类中,一般形式是:

```
public class AppName{
      public static void main(String args[])
      {
      … // 代码
      }
}
```

(2) Java 没有全局变量

在 Java 程序中,不能在类外面定义全局变量,只能通过在一个类中定义公用静态变量来实现全局变量。例如:

```
class GlobalVar {
      public static GlobalVarName;    // 全局变量定义
}
```

因为 public static 成员是一种类属成员变量,只要定义了类,其中的类属成员变量就分配空间,而不需要声明类的对象,使得其他类可以访问和操作该变量。

可见，在 Java 中，对全局变量进行了更好的封装。而在 C++中，不依赖于任何类的不加封装的全局变量往往会导致系统崩溃。

(3) Java 没有结构和联合

在 C++中，为了保持和 C 的兼容，继续支持结构(struct)和联合(union)。但是，在 Java 中，则完全摒弃了这些面向过程时代的概念。

(4) 字符串不再是字符数组

在 C 和 C++中，字符串操作往往会导致许多内存问题，如内存非法操作、内存泄漏等。因为字符串 char *s 和不定界的字符数组 char s[]是等价的。但两者只是为变量 s 分配一个指向字符串的指针，存储字符串内容的内存需要申请和释放。

在 Java 中，字符串和字符数组已经被分开了。字符串是一个完全意义上的对象，需要用 String 类来定义。

(5) Java 用包(Package)来分解命名空间

在大型的软件工程中，怎样保证程序员之间命名的类不重名呢？如何避免供应商提供的类和程序员自己命名的类不重名呢？虽然有许多方法可以避免重名，但是如果在发现问题以前，工程已经启动了，要修改这些重名问题，就变得非常麻烦。

在 Java 中，引入 Package 概念来解决上述问题。Package 有效地通过集合类来划分命名空间，在不同包内的类可以同名，但不会引起混乱。

Java 并没有彻底解决命名冲突的问题，扩展基类可能引起派生类的冲突。例如，用户派生一个类，增加一个方法 foo。如果以后供应商提供新的版本，在新类中也包含了 foo 方法，冲突就出现了。

(6) Java 没有独立的头文件

在 C++中，每一个.cpp 实现文件都对应一个.h 头文件，在头文件中，往往包含了.cpp 文件中用到的类的定义。在 Java 中，关于类的一切东西(属性和方法)都被放到一个单独的文件中，类中方法的实现必须在定义的过程中同时进行。

因为类方法的实现代码必须在方法定义时完成，但是一个函数的代码往往是几十行或者几百行的程序代码，这样就使得阅读类很难一下子就看到一个类的全貌。Java 的设计者已经考虑到了这个问题，为此，在 JDK 中提供了两个工具来补偿，Javap 来打印类标识，Javadoc 为源代码提供标准的 HTML 文档。

(7) 数据类型

在 C/C++中，对于不同的平台，编译器对简单数据类型 int、float 等分配不同长度的内存空间，例如 int 在 IBM PC 中为 16bit，在 VAX-11 中为 32bit，这导致了代码的不可移植性。但是在 Java 中，对于这些基本数据类型，总是分配固定长度的空间，int 总是 32bit，这就可以保证 Java 的平台无关性。

在 C/C++中通过指针可以进行强制类型转换，这往往会带来不安全性。在 Java 中，有严格的类型相容性检查。另外，在 Java 中，没有模板类，而 C++中的模板类(参数化类，即形式参数对应的实际参数是数据类型)可以有效地简化程序代码的编写，但不能减少可执行代码的长度。这就意味着，在 Java 中，只能靠相似代码的手工复制和修改。

(8) 常量修饰符 const 的使用限制

在 C++中，const 常量修饰符有着重要的应用，它对于提高代码质量起到了积极的作用。

例如,用户可以声明函数参数或者函数的返回值为 const 类型,这样可以有效地防止在函数内部对函数参数的不正当修改或者对返回值的修改。另外,可以将类的一个成员函数声明为 const,表明该方法不能修改它操作的任何对象。

在 Java 中,支持常量操作符、只读变量,通过 final 关键字实现。但是,Java 没有提供一种机制,使得一个变量在参数传递或者返回的过程中只读,或者定义一个不能修改操作对象的常量方法。上述的省略,为 Java 程序带来了一个可能引起不正当修改错误的隐患。

另外,在 Java 中,宏也不再被支持。

### 6.2.2 Java 程序设计语言

Java 程序设计语言和 C/C++等一般的程序设计语言类似,都包含基本符号、数据、数据类型、表达式、流程控制、类与对象等程序设计的概念。此外,Java 程序设计语言还包含了一些自身的特点,例如接口、包、小程序等。

#### 1. 基本符号

任何一门程序设计语言都有其自身的字符集和基本符号,它们按照语法构成语言的语句,由语句再构成程序。

(1) 基本字符

Java 语言的基本字符有字母(a、b、…、z,A、B、…、Z)、数字(0、1、…、9)和特殊符号(+、−、*、/、<、=、>等)。

(2) 保留字(关键字)

保留字是由字母构成的具有固定含义的单词,保留字通常用于程序语句中,例如,if 代表条件语句,while 代表循环语句等。

(3) 标识符

在 Java 程序中,标识符是表示数据类型、类、接口、变量、方法(函数)等名称的符号。标识符是以字母、下划线或 $ 符号开头的字母、数字、下划线组成的字符序列。标识符的长度任意,区分大小写,用户自定义的标识符不能使用保留字或标准标识符。

标识符和保留字在形式上非常类似,但含义不同,标识符标识的是一个名称,例如,int 就是一个标识符,表示整数类型。

(4) 注释

为了增加程序的可读性和便于程序的维护,往往在重要的位置添加注释。注释原则上可以出现在程序中的任何位置,但是如果使注释与程序的结构配合起来,则效果更好。注释一般分为序言性注释和描述性注释两种类型。

序言性注释出现在模块的首部,其内容一般包括有关模块功能的说明和界面描述,包括调用语句格式、所有参数的解释和该模块需调用的其他模块名等,还包括一些重要变量的使用、限制及其他信息。描述性注释嵌在程序之中,描述性注释又有功能性的和状态性的。功能性注释说明程序段的功能,通常可放在程序段之前;状态性注释说明数据的状态,通常可放在程序段之后。

Java 程序中的注释有以下三种形式:

① 以"//"开头的单行注释。

② 以"/＊ …… ＊/"标记的块注释。

③ 以"/＊＊ …… ＊/"标记的文档注释，即"/＊＊"和"＊/"之间的所有内容(可能占多行)都是注释内容。

## 2. 数据

程序是对数据的处理，数据分成常量和变量，无论是常量还是变量，每个数据都有一个相应的数据类型。

**常量**是指在程序执行过程中，其值不发生变化的量。根据类型不同，常量可分为整型常量(如 123、－15)、实型(浮点型)常量(如 12.34)、字符常量(如 'x')、布尔常量(如 true)等。

对于字符数据，还有一些特殊字符，在程序中具有特殊含义，见表 6-1。

表 6-1 特殊字符

| 字 符 | 含 义 | 字 符 | 含 义 | 字 符 | 含 义 |
|---|---|---|---|---|---|
| \n | 换行 | \f | 进纸 | \ddd | 八进制表示法 |
| \t | 水平制表符 | \\ | 表示"\" | \xdd | 十六进制表示法 |
| \b | Backspace | \' | 单引号 | \udddd | 十六进制表示 unicode 字符 |
| \r | 回车 | \" | 双引号 | | |

所谓**变量**，是指在程序执行过程中，其值发生变化的量。每一个变量都有一个变量名，变量名是一个用户自定义标识符，每个变量都有一个数据类型。类型决定变量在内存中所占空间的大小，同时决定变量的取值范围和操作运算。

和 C/C++不同，在 Java 中，没有全局变量，所有的变量都被封装在类中，成为类的成员变量。但是，可以通过静态成员变量实现类似的全局变量功能。

## 3. 数据类型

Java 的数据类型可分为基本数据类型和构造数据类型。

(1) 基本数据类型

基本数据类型包括整型(byte、short、int、long)、浮点型(float)、双精度型(double)、布尔型(boolean)和字符型(char)。

(2) 构造类型

在 Java 中，没有 C/C++中面向过程的结构和联合。Java 中的构造类型是用类来描述的，如数组、字符串、对象、类等。

类型用于说明类中的成员变量或成员函数的返回值。一般形式是：

**＜类型名＞ ＜变量$_1$，变量$_2$，……，变量$_n$＞；**

## 4. 表达式

表达式是由常量、变量、函数、运算符以及括号连接而成的式子。常用的表达式运算符见表 6-2。

表 6-2 表达式运算符

| 运算符 | 功 能 | 运算符 | 功 能 | 运算符 | 功 能 | 运算符 | 功 能 |
|---|---|---|---|---|---|---|---|
| ＝ | 赋值运算 | & | 按位与 | ! | 逻辑非 | ＜ | 小于 |
| ＋ | 加运算 | \| | 按位或 | && | 逻辑与 | ＜＝ | 小于等于 |
| － | 减运算 | ～ | 按位取反 | \|\| | 逻辑或 | ＝＝ | 等于 |
| * | 乘运算 | ^ | 按位异或 | ?: | 条件运算 | !＝ | 不等于 |
| / | 除运算 | ＜＜ | 按位左移 | ＞ | 大于 | ＋＋ | 自加 |
| % | 取模运算 | ＞＞ | 按位右移 | ＞＝ | 大于等于 | －－ | 自减 |

在表达式中,运算符的优先级和一般的程序设计语言相似。根据表达式运算结果的不同,表达式又分为算术表达式、字符表达式、逻辑表达式等。

在 Java 表达式中规定,整型、实型、字符型数据可以混合运算。运算过程中,不同类型的数据会自动转换为同一类型,然后再运算。自动转换按低级类型转换成高级类型数据的规则,转换规则为(其中 op 是运算符,如＋、－等):

(1) (byte 或 short)op int → int

(2) (byte 或 short 或 int)op long → long

(3) (byte 或 short 或 int 或 long)op float → float

(4) (byte 或 short 或 int 或 long 或 float)op double → double

(5) char op int → int

如果要把高级类型转换成低级类型,要通过强制类型转换完成,一般形式是:

**(类型名)表达式**

上述形式将把表达式的计算结果强行转化为前面类型名所指定的数据类型。

### 5. 语句

所有的程序设计语言,其语句部分都是类似的,都包括:赋值语句、输入/输出语句、分支语句和重复语句等类型。为了语法上的需要,不同的语句还可以组合成逻辑上的一个语句,即构成一个语句块。

(1) 赋值语句

在 Java 中,赋值语句由赋值表达式加一个分号构成,一般形式为:

**＜变量名＞ = ＜表达式＞;**

赋值语句的逻辑功能是将右边表达式的值赋给左边的变量。

(2) 输入/输出语句

在 Java 中,变量的输入和表达式的输出是通过 java. lang. System 类中的 System. in 和 System. out 方法完成的,它提供从键盘输入和从终端输出的方法。具体的使用在后续的相应类中介绍。

(3) 分支语句

分支语句又称选择语句,有三种形式:

① if 语句

一般形式是:

```
if  (<表达式>)
    <语句>;
```

if 语句首先计算表达式的值，若结果为 True，则执行语句部分，否则执行 if 语句下面的语句。注意，表达式必须要用圆括号括起来。

② if-else 语句

一般形式是：

```
if(<表达式>)
   <语句 1>;
else
   <语句 2>;
```

首先计算表达式的值，若为 true，则执行语句 1，否则，执行语句 2。if-else 语句可以按照需要进行嵌套，嵌套会降低程序的可读性。

③ switch 语句

switch 语句提供了 if-else 语句多层嵌套结构的一个变通形式，可以从多个语句块中选择执行其中的一个。switch 语句提供的功能和 if-else 语句类似，但是可以使代码更加简练易读。一般形式为：

```
switch ( <表达式>) {
    case 常量 1:
        语句 1
        [ break; ]
    case 常量 2:
        语句 2
        [ break; ]
    …
    case 常量 n:
        语句 n
        [ break; ]
    [default:
        默认处理语句
    ]
}
```

在 switch 语句中，表达式为数值或字符表达式，首先要计算表达式的值，表达式的值将与结构中每个 case 的常量表达式的值比较。如果匹配，则执行与该 case 关联的语句。break 语句将使流程退出 switch 语句块，从而执行 switch 语句后面的语句，否则，将执行下面的 case 判断。

(4) 循环控制语句

当部分语句需要反复执行时，需要循环语句。循环语句总是由循环体和循环终止条件两部分组成，在 Java 语言中，循环语句有 while、do while 和 for 三种形式。

① while 语句

一般形式：

```
while(<表达式>)
    <语句>;
```

While 语句首先计算表达式的值,如果为真,则执行循环体。然后无条件地返回 while 语句的开始,继续计算表达式的值。如果条件表达式结果为 false,则结束循环,执行下面的语句。

② do-while 语句

一般形式为:

```
do {
   语句(循环体)
} while(<表达式>);
```

直接执行循环体,然后计算表达式的值,如为 true,无条件地返回循环体的第一条语句,继续执行循环体,计算条件表达式的值。如果条件表达式结果为 false,则结束循环,执行 do…while 语句下面的语句。

③ for 语句

一般形式为:

```
for (表达式 1; 表达式 2; 表达式 3)
     <语句>;
```

执行过程如下:

Step1:计算表达式 1。

Step2:计算表达式 2,如果为真,则执行循环体,然后转 Step3;否则,结束循环。

Step3:计算表达式 3。

Step4:无条件转 Step2。

Step5:循环结束,执行 for 语句下面的语句。

在三种循环语句中,任何一种语句都可以写成另外两种语句的形式。也就是说,只需一种形式的循环语句就可以表达程序的重复结构。具体使用哪一种,应该根据三种语句各自的特点和编程人员的编程习惯而定。

另外,循环体的内部也可以包含循环语句,即循环的嵌套,构成多重循环。

(5) break 语句

break 语句用于跳出最近的循环(语句块{…}),一般形式为:

```
break;
```

例如:

```
for (i = 0,p = 1;true;i++) {
      p * = 2;
      if (p>1024) break;
}
```

上述代码用于计算 $p=2^i$,当 $p$ 大于 1024 时,退出循环,计算结束。

(6) continue 语句

break 语句用于结束本次循环,即跳过本次循环中下面尚未执行的语句,接着进入下一次是否执行循环的判断,一般形式为:

```
continue;
```

(7) 语句块

在上述的语句中,许多语句的语法中是一个语句,例如 while 语句、for 语句,其循环体是一个语句。但是,在大多数情况下,循环体的功能通常是需要多个语句才能完成的,由于在 C/C++和 Java 中,分号(;)是语句的分隔符,此时需要用“{”和“}”将多个语句括起来,构成逻辑上的一个语句,这就是语句块。

例如,计算一个整数的位数,程序代码如下:

```
width = 0;
while (num! = 0)
{
     width++;
     num = num/10;
}
```

在上述代码中,循环语句 while 的循环体就必须使用语句块,里面包含了两条简单的程序语句。

## 6.2.3　类与对象

在 20 世纪 90 年代以前,自顶向下逐步求精的结构化程序设计是软件开发的主要方法,直到现在,这种结构化的程序设计思想仍然被广泛地采用。Pascal、C、Basic、Fortran 等高级语言很好地实现了结构化编程的思想,通过过程和函数(又称子程序),把一个复杂的问题分解成几个相对简单的子问题,如果子问题还比较复杂,再继续划分,最后将划分后的每个问题用过程和函数来实现。

20 世纪 90 年代以后,面向对象技术对人们近半个世纪以来的软件开发思想产生深刻的变革。这一技术强调利用软件对象进行软件开发,它将自然界中的物理对象和软件对象相对应,建立了类和对象的概念。由于客观世界的实体和软件结构的对象一一对应,从而增加了软件系统的可扩展性和可维护性。面向对象技术将自然界中的物理对象和软件对象对应起来,在传统的数据结构基础上加入了成员函数(方法)的概念,从而赋予对象以动作。

### 1. 类与对象的概念

**类**(class)是包含数据和处理这些数据的过程的数据结构。我们可以将类看成是和 int、float 等基本数据类型一样的数据类型,用它来创建数据对象,它指定了相应内存区域的处理和解释规则。

**对象**(object)是用类来声明的数据结构,如果将类比作数据类型,对象就是相应数据类型的变量。对象是类的实例,占据确定的内存空间。

### 2. 类的定义

在 Java 中,用户可以定义一个基类,也可以从别的类进行派生(extends),或者通过实现(implements)一个或多个接口来定义一个新的类。类定义的一般形式是:

**[修饰符] class <类名> [extends <父类名>] [implements <接口名列表>]**

```
{
    成员变量声明;
    成员函数定义;
}
```

其中,修饰符决定类的类型,有四种:

- abstract:抽象类,抽象类必须包含至少一个抽象成员函数。抽象类不能创建对象,需要用其派生类创建。例如,可以定义一个图形类 CFigure,将其定义为抽象类,然后从其派生具体的图形类,例如三角形类 CTriangle、矩形类 CRectangle 等。
- final:最终类,说明一个类不能再派生子类。
- public:能被其他类访问的公有类,在其他包(package)里要使用该类,需要先用 import 导入,否则只能在它定义的 package 里使用。
- synchronicable:表示所有类的成员函数都是同步的。

上述修饰符可以同时出现两个或以上,但修饰符 abstract 和 final 不能同时使用。

class 为关键字,表明接下来是类的定义,类名是一个用户自定义的标识符。

如果是派生类,需要通过 extends 给出父类,如果实现接口(interface),需要通过 implents 给出接口名,接口可以是一个或多个。

花括号内说明类的成员变量和成员函数,又称类的属性(attribute)和方法(method)。

成员变量定义的一般形式是:

```
[修饰符] <类型> <成员变量列表>;
```

其中修饰符为类成员的访问级别声明符号,可以是 private、public、protected。还可以添加的修饰符有 final(最终,初始化后不能再改变其值的变量)、volatile(多线程变量)。这些访问级别可以按任何顺序出现,也可以多次出现。在两个访问级别声明符号之间的成员具有相同的访问控制级别。

如果成员变量包含修饰符 static,此变量称为类变量,不加 static 的变量称为对象变量。如果变量说明时缺省修饰符,则它可被同一包中的所有类访问。

成员函数又称类的方法(method),是类中定义的可对类进行操作的函数模块。成员函数定义的一般形式为:

```
[修饰符] 返回值类型 <方法名>([形式参数列表])[throws 异常列表]
{
// 函数体(Java 程序代码)
}
```

其中修饰符可以是 private、public、protected 和 final(最终,不能由子类改变的方法)、abstract(抽象方法,无方法体)、synchronized(线程同步的方法)、native(本机方法)。修饰符可以有一个或多个。另外,还可以有 static 来声明静态成员函数,表明此方法为类方法,无 static 修饰的方法为对象方法。

“形式参数列表”给出函数的形式参数及类型,形参之间用逗号分隔。当方法没有形参时圆括号内为空。

### 3. 构造函数

在面向对象技术中,对象是类的实例,每个对象必须按照类的定义来创建,这种机制是

通过类的构造函数来实现的。构造函数(constructor)是一种特殊的成员函数,用来在内存中建立具体的对象。构造函数必须申请必要的内存空间,将内存转化为具体的对象,初始化成员变量等。构造函数的名称和类名称相同,一个类可以拥有几个带有不同参数的构造函数。构造函数没有返回类型和返回值。

和一般的函数不同,构造函数不是由用户显式调用(call)的,它是通过编译器来调用的,称为激活(invoke)。当通过 new 创建一个类的对象时,相应的构造函数则被激活,来完成内存对象的初始化操作等。

在使用构造函数时,需要注意以下若干情况:

(1) 构造函数不能描述为 const 和 volatile。

(2) 构造函数不能是 static,因为构造函数需要初始化类的成员变量,但静态构造函数不能访问成员变量。

(3) 构造函数不能被继承。当一个没有构造函数的类从一个含有构造函数的类派生时,将写它自己的构造函数。

(4) 构造函数不能有返回值,也不可以是 void。

(5) 定义一个类时,必须明确定义除缺省构造函数和复制构造函数以外的所有其他构造函数。缺省构造函数是不含有参数的构造函数,可以省略不写。

#### 4. 创建对象

当定义类后,可以使用 new 来创建对象,一般形式是:

```
对象变量 = new 类(参数);
```

如果类的构造函数带有参数,可以通过 new 中类名后面的参数来激活相应的构造函数。

#### 5. Java 程序与 main()方法

所有的程序都应该有一个主程序,它是程序开始执行的入口。在 Java 程序中,通常定义多个类,但只能有一个 public 类。在这些类中必须有一个类包含 main()方法,如果包含 main 方法的类是公有类(用 public 修饰的类),那么文件名必须是这个类的类名,如果 main 方法所在的类没有用 public 修饰,那么文件名可以任意命名,不一定要和任何类名一样。

**【例 6-1】** 类的定义举例。

在下面的例子中,定义了一个三角形图形类 CTriangle,包含三个私有成员变量,存储三角形的三条边,一个构造函数,一个计算三角形面积的成员函数和一个输出成员函数,文档名为 exa01.java,代码清单如下:

```
class CTriangle
{
  private double a,b,c,t;
  public double s = 0;
  CTriangle(double x,double y,double z)
  {
    a = x; b = y; c = z;
  }
```

```
    //计算三角形的面积
    public double Area()
    {
      t = (a + b + c)/2;
      s = Math.sqrt(t * (t - a) * (t - b) * (t - c));
      return s;
    }
    public void out1()
    {
      Area();
      System.out.println("Area = " + s);
    }
  }
  class MyTest01
  {
    public static void main(String[] args)
    {
      CTriangle myobj  =  new CTriangle(10,20,15);
      myobj.out1();
    }
  }
```

在DOS提示符下,执行javac exa01.java命令,编译,生成每个类的.class文件,即CTriangle.class和MyTest01.class。执行java MyTest01,运行上述程序,输出结果如图6-2所示。




图6-2　Java程序的编译和运行

**【例6-2】** 类的构造函数激活次序举例。

在下面的例子中,定义了三个类A、B、C,其中C中包含两个对象成员,来展示对象构造函数的激活次序,文档名为exa02.java,内容如下:

```
class A {
    int x,y;
    A(int a,int b)
    {
```

```
        x = a; y = b;
      }
      A() // 默认构造函数
      {
        x = 0; y = 0;
        System.out.println("A constructor\n");
      }
  }
  class B {
      B()
      {
        System.out.println("B constructor\n");
      }
  }
  class C {
      public A a = new A();
      public B b = new B();
      C()
      {
        System.out.println("C constructor\n");
      }
  }

  class MyTest02
  {
    public static void main(String[] args)
    {
      C myObj = new C();
    }
  }
```

执行 java MyTest02,输出结果为:

```
A constructor
B constructor
C constructor
```

上述输出结果表明,如果一个类含有成员对象,则在创建类的对象时,将先创建类的成员对象,最后才是类的对象本身。

### 6. 封装和抽象

在面向对象技术中,一个主要的目标就是对象的封装和抽象。封装(Encapsulation)是指对象可以拥有私有元素,将内部细节隐藏起来的能力。封装将对象封闭起来,管理着对象的内部状态。而抽象则和对象的外部状态紧密相关,它通常用来描述对象所表示的具体概念、对象所完成的任务以及处理对象的外部接口。抽象处理的是对象的可见外部特征。

在 C 中,通过关键词 static 可以实现有限的封装。当一个变量在一个函数内部被说明成 static 形式时,该变量就只在函数中存在,并且只在函数内部有效。另外,一个全程变量被说明成 static 形式时,该变量只在其所在的文件有效,这样可以避免不同的文件中全局变

量的重名。

在 C++ 和 Java 等面向对象的程序设计语言中,类的每一个成员都被说明成 public、private 和 protected 型,用这些关键词来实现数据的抽象和封装。

(1) 关键词 public

类中所有 public 成员构成类的接口,它们是类的抽象性的表现。出于简单、经济和安全的愿望,类的公开元素越少越好。但是,类又必须和外部打交道,public 成员是不可缺少的。

(2) 关键词 private

类中的 private 成员只能被类的成员函数、友元类或外部友元函数访问。从而实现类的封装性。在默认状态下,所有的成员都是私有的。

(3) 关键词 protected

在面向对象技术中,派生是类的重要性质。类的 private 成员将不能被派生类中的成员访问,这就大大限制了类的灵活性。类的 protected 成员可以被类的派生类成员访问。

类成员访问规则如图 6-3 所示。

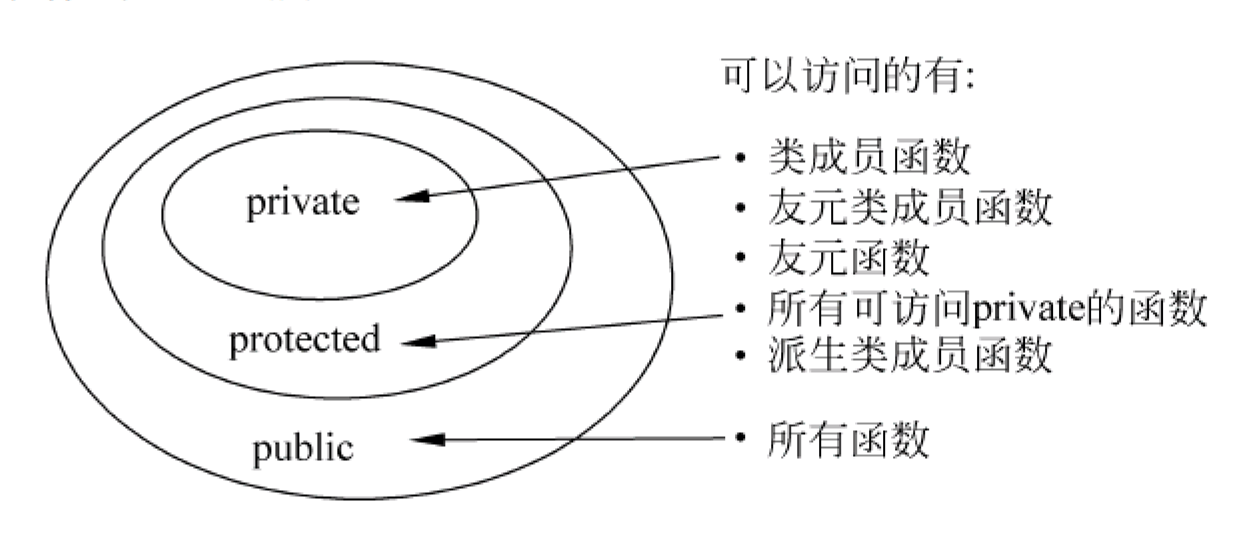



图 6-3 类成员访问规则

### 7. 静态成员

支持封装和抽象的另外机制还包括关键词 static。当一个成员被说明成 static 时,则该成员在程序中只有一个复制存在,而不是在每个对象中都有一个复制。所有的对象共享类中的静态成员。在一些面向对象的程序设计语言中,静态成员被称为类变量,或类属变量。

例如,定义一个含有静态成员变量的类如下:

```
class CS {
      static int a;
      int b,c,d;
};
CS objs[3];
```

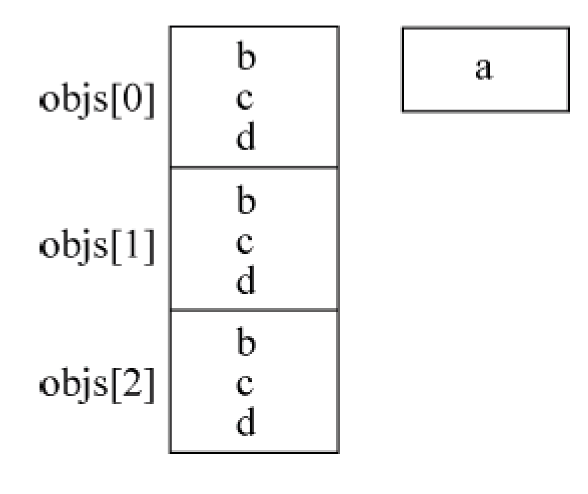



图 6-4 类 CS 的成员

在类 CS 中,包含四个成员变量,其中成员变量 a 为静态成员变量。数组 objs 包含三个 CS 类对象,由于 CS 类包含一个静态成员变量 a,因此三个 CS 对象共享一个静态成员变量 a,每个对象都包含自己的普通成员变量 b、c、d,如图 6-4 所示。

还可以把一个成员函数说明成 static,这意味着在没有任何该类对象的情况下,它仍然可以被执行。正因如此,静态成员函数能够完成不需要任何对象成员变量的操作。静态成员变量和

静态成员函数为类存储和管理属于整个类而不是个别对象的信息提供了一种方法。

8. 类的继承性与派生类

从现存的对象出发建立一种新的对象类型，使它继承原对象的特点和功能，这就是对象的继承性。继承是许多层次对象的自然描述。从现存类派生出新类时，现存类称为**基类**(base class)，派生出的新类称为派生类。派生类可以对基类作如下变化：

- 增加新的成员变量。
- 增加新的成员函数。
- 重新定义已有的成员函数，即，子类可以对父类的方法覆盖(overriding)或重载(overloading)。所谓覆盖，是指在子类中定义了与父类中同名的函数，这时将根据当前的对象类型执行子类中的代码，即父类中的代码被覆盖。所谓方法重载是子类与父类的方法同名，但是形式参数不同时，将重新实现该方法。
- 改变现有成员函数的属性。

派生类不能删除基类的成员变量和成员函数，实际上派生类往往是基类的扩充，是一种具体化和完善的过程。

类的派生创建了一个类族，派生类的对象也是基类的一个对象，它可用在基类对象可被使用的任何地方。可用多态成员函数来调整这种关系，以使得派生类在某些地方和它的基类一致，而在另外一些方面表现出其自身的行为特征。

类的派生是一种演化过程，即通过扩展、更改和特殊化从一个已知类出发来建立一个新的类。类的派生建立了一个具有共同关键特性的类族，从而实现代码的重用。假设从一个已知基类 A 建立一个派生类 B，一般形式为：

```
class B extends A
{
    // 派生类 B 的成员说明
};
```

读作"类 B 由 A 派生"，它告诉编译器类 B 是一种 A，对基类 A 所作的修改和添加在括号内给出。在派生类对象中，编译器在内存中总是先放入基类的成员，后面是派生类独有的成员，如图 6-5 所示。

| 基类成员 |
| --- |
| 派生类<br>特有成员 |

图 6-5　派生类对象内存结构

对于一个派生类，对于其父类中的成员，有特定的成员的访问规则，见表 6-3。

表 6-3　基类和派生类中成员的访问特性

| 基类 class A {…} | 派生类 class B extends A {…} | 基类 class A {…} | 派生类 class B extends A {…} |
| --- | --- | --- | --- |
| private 成员 | 不可访问 | public 成员 | 公有 |
| protected 成员 | 保护 | 不可访问成员 | 不可访问 |

所谓不可访问成员，是一个派生类中的概念。一个根类(不是从其他类派生出来的类)中不存在不可访问的成员。但是在派生类中，不可访问是存在的，如根类中的 private 成员，就构成派生类的不可访问成员。如果再从派生类派生新的类，则派生类的不可访问成员和

私有成员就构成了它的派生类的不可访问成员,依此类推。

**【例 6-3】** 对象成员的访问举例。

下述例子演示了类及其派生类中成员的访问规则,文档名为 exa03.java,代码如下:

```
class B {
    private int x,y;
    protected int z;
    B()                        // 构造函数 1,默认构造函数
    {
      x = 0; y = 0; z = 0;
      System.out.println("A constructor\n");
    }
    B(int a,int b)             // 构造函数 2
    {
      x = a; y = b;
      z = x + y;
    }
}
class D extends B{
    D(int x,int y)             // 构造函数
    {
      System.out.println("D constructor\n");
    }
    public void out()
    {
      System.out.println("D public out Access class B:z = " + z); // 访问父类的 protected 成员
    }
}
class MyTest03
{
   public static void main(String[] args)
   {
     D myd =  new D(10,20);    // 创建派生类对象
     myd.out();                // 外部访问
   }
}
```

执行 java MyTest03,运行结果为:

```
A constructor
D constructor
D public out Access class B:z = 0
```

从上面的例子可以看出,创建一个派生类的对象,首先执行父类的构造函数 1,然后再执行本身的构造函数。在派生类中,可以访问父类的 protected 成员。

问题:在一个派生类中,如何执行父类的带有参数的构造函数呢?在 C++中,可以在派生类的构造函数首部写出要调用的父类的构造函数,例如:

```
class Derive::public Base {
```

```
    public :
      Derive(int m = 0, int n = 0):Base(m) {
      …
      }
…
}
```

上述代码表示，在创建派生类 Derive 对象时，调用的父类的构造函数是 Base(m)，而不是默认构造函数。

### 9. 多态性和抽象类

在传统的程序设计语言中，函数是不能重名的。但是，在 C++/Java 等面向对象的程序设计语言中，在一个类中，可以定义多个函数名相同，但参数不同的函数；或者，在一个派生类中，可以定义一个和父类的某个函数重名的函数。这样，当相同的操作作用于不同类的对象时，就可能得到不同的结果，这就是类的多态性(Polymorphism)。

在 C++/Java 中，成员函数的多态性有方法的重载(Overloading)和重写(Overriding)两种形式。重载 Overloading 是一个类中多态性的一种表现，如果在一个类中定义了多个同名的方法，它们或有不同的参数个数或有不同的参数类型，则称为方法的重载(Overloading)。重写 Overriding 是父类与子类之间多态性的一种表现，如果在子类中定义某方法与其父类具有相同的名称和参数，我们说该方法被重写(Overriding)。子类的对象使用这个方法时，将调用子类中的定义，对它而言，父类中的定义如同被"屏蔽"了。

根据成员函数的重载和重写特性，多态就分为编译时多态和运行时多态两种情况。所谓编译时多态主要包括操作符重载和函数(方法)重载。在程序编译时，系统根据传递的参数的个数和参数类型等信息决定要连接的函数。运行时多态是指直到系统运行时才根据操作对象的类来决定执行何种操作，即执行父类还是派生类中的方法。

运行时多态极大地增强了类的抽象能力，在 C++中，运行时多态是通过在父类中使用虚函数(Virtual)来实现的。在 Java 里没有虚函数的概念，有抽象函数，但抽象函数并不是 C++虚函数在 Java 里的等价物，两者是有区别的。

下面是 C++和 Java 关于多态性和抽象类概念的对比。

| C++ | | Java |
|---|---|---|
| 虚函数 | -------- | 普通函数 |
| 纯虚函数 | -------- | 抽象函数 |
| 抽象类 | -------- | 抽象类 |
| 虚基类 | -------- | 接口 |

在 Java 中，抽象函数必须定义在抽象类里面，而且没有方法体。例如：

```
public abstract TestAbstract {
       public abstract void say(); // 抽象函数,没有函数体
}
```

所谓抽象类，就是用 abstract 修饰符说明的类。当一个类包含一个 abstract 成员函数时，必须将这个类定义为 abstract 类。然而，并不是抽象类的所有成员函数都是 abstract 的，抽象类不能有私有成员函数和静态成员函数。

【例 6-4】 编译时多态性举例。

定义一个类,包含两个成员函数 Area,分别用于计算矩形和圆的面积,文档名为 exa04.java,代码如下:

```
class CFigure
{
    public double s = 0;
    public double Area(double a,double b)
    {
      s = a * b;
      return s;      //计算矩形的面积
    }
    public double Area(double r)
    {
      s = 3.14 * r * r;
      return s;      //计算圆的面积
   }
}
class MyTest04
{
    public static void main(String[] args)
    {
      CFigure myobj  =  new CFigure();
      System.out.println("Rectangle area = " + myobj.Area(10,20));
      System.out.println("Circle area = " + myobj.Area(10));
    }
}
```

在 DOS 提示符下,执行 javac exa04.java 命令,编译,生成每个类的.class 文件;然后执行 java MyTest04 运行上述程序,输出结果如下:

```
Rectangle Area  = 200.0
Circle Area  = 314.0
```

在上述代码中,CFigure 类包含两个同名的成员函数 Area,但参数不同。编译器在编译时,会根据 Area 中的实际参数,确定需要调用哪一个 Area 函数,即实现编译时多态。

【例 6-5】 运行时多态性举例。

下面例子演示了 Java 中类的运行时多态,文档名为 CMyFigure.java,内容如下:

```
public abstract class CMyFigure                 // 定义抽象类
{
    //public abstract void Draw();
    public abstract void Area();                // 声明一个抽象函数,无函数体
}
class CRectangle extends CMyFigure              // 定义一个派生类 CRectangle
{
    float height,width;
    float s;
    CRectangle(int x,int y)
    {
```

```
        height = x;
        width  = y;
    }
    public void Area()                    // Overriding 父类中的抽象函数
    {
        s = height * width;
        System.out.println("The Area of the Rectangle = " + s);
    }
}

class CCircle extends CMyFigure           // 定义一个派生类 CCircle
{
    float r,s;
    CCircle(float x)
    {
        r = x;
    }
    public void Area()                    // Overriding 父类中的抽象函数
    {
        s = (float)3.14 * r * r;
        System.out.println("The Area of the Circle = " + s);
    }
}

class MyTest05
{
    public static void main(String[] args)
    {
        CMyFigure[ ] objs  =  new CMyFigure[2]; // 创建一个定长数组
        objs[0]  =  new CRectangle(10,20);
        objs[1]  =  new CCircle(10);
        for (int i = 0; i<2; i++)
            objs[i].Area();
    }
}
```

在上述代码中，首先创建了一个 CFigure 类型的数组，但在程序运行时实际存储的是派生类 CRectangle 和 CCircle 对象。在接下来的 for 循环中，则根据具体的对象类，分别执行了 CRectangle 和 CCircle 中的 Area 中的成员函数，而不是 CFigure 的 Area 函数，这是一种运行时多态。上述代码展示了运行时多态的强大功能，使得程序代码更加精致。

在 DOS 提示符下，执行 javac CMyFigure.java 命令，编译，生成每个类的.class 文件；然后执行 java MyTest05 运行上述程序，输出结果如下：

```
The Area of the Rectangle = 200.0
The Area of the Circle = 314.0
```

特别注意，运行时输入 java MyTest05，即包含 main 的类，而不是 java CMyfigure，否则将产生运行错误：Exception in thread "main" java.lang.NoSuchMethodError：main。

最后，需要说明的是，在调试 Java 程序时，经常是编译通过了，但运行时出现下列错误

提示：Exception in thread "main" java.lang.NoSuchMethodError：main。产生上述错误的原因比较复杂，可以从以下几个方面进行排查：

(1) 每个Java文件有且只能有一个公有类，即public类，文件名必须和这个公有类的类名大小写完全一样。在要运行的类中有且只能有一个public static void main(String[] args)方法，这个方法就是主程序。运行一个Java程序，应该是在java命令后跟包含main函数的类名，而不是Java程序的文件名，即java ＜包含main方法的java类＞，类名后面不能有.class等扩展名。

(2) 如果在编译或运行时出现问题，还可能是类的路径问题。我们来看一下Java程序的运行过程，首先通过javac命令，将java程序编译生成.class文件，即虚拟机要执行的代码，称之为字节码(bytecode)。然后，通过Java命令，即通过Java虚拟机来解释运行class文件。虚拟机通过classloader来装载这些字节码，即通常意义上的类。问题是，classloader从哪里知道Java本身的类库及用户自己的类在什么地方呢？要解决这个问题，通常是通过系统环境变量中的类路径(classpath)来设置的。

因此，当出现编译或运行错误时，还应该检查类路径设置是否正确，具体方法是：在DOS提示符下输入set classpath，即可显示当前的路径设置，应该包含一个当前路径项目，即"."，其次就是Java的安装路径。

### 6.2.4 接口

在一个复杂的面向对象的系统中，实现一个有更多方法的新类是经常遇到的。当一个类需要从多个基类派生时，派生类将继承多个基类的特征，在C++中，这样的机制称为多重继承。在面向对象的程序设计中，继承关系一直存在很多的争议，特别是多重继承。

Java没有多重继承，可以通过接口来实现相应的功能。在Java中，所谓接口(Interface)是一组没有给出实现细节的操作(方法)的集合。它需要别的类来实现接口给出的每一个方法，一个类可以实现一个或多个接口。如果一个类实现了某个接口，就相当于声明我能够完成某项工作。在许多情况下，接口继承(implements关系)比实现继承(extends关系)更有优势。

#### 1. 创建接口

在Java中，可以定义一个接口，也可以从一个接口或多个接口来扩展一个接口，这和类的定义类似。接口定义的一般形式是：

```
public interface <Interfacename> [extends <Superinterface列表>]
{
    成员变量声明(常量);
    成员函数(方法)声明;
}
```

在接口定义中，public指示了接口可以在任何的包中任何的类中使用。如果你没有指定接口为public，那么接口就只能在定义接口的包中的类中使用。一个接口可以扩展另外的接口，这跟类可以扩展一样。但是，类只能扩展一个另外的类，而接口可以扩展任意个接

口。Superinterface 列表列出所有的被扩展的接口，以逗号分隔。

接口可以包含常量声明以及方法声明。所有定义在接口中的常量可以是 public、static 和 final。定义在接口中的成员变量不能使用 transient、volatile 或者 synchronized 修饰符。同样也不能在声明接口的成员的时候使用 private 和 protected 修饰符。

接口中的方法声明后紧跟着一个分号，因为接口中的方法不需要给出具体的实现代码。因此，所有定义在接口中的方法可以隐含地为 public abstact 方法。

### 2. 接口和类

接口的定义和类的定义类似，但是接口不是一个类，而是对符合接口要求的类的一套规范。接口说明了实现接口的类该做什么而不指定如何去做，一个类可以实现一个或多个接口。

实现一个接口需要两个步骤：

(1) 声明类需要实现的接口，声明一个类实现一个接口需要使用 implements 关键字。

(2) 提供接口中的所有方法的定义。为了使用接口，需要编写执行接口的类。一个类实现了某个接口，即这个类实现了在接口中声明的所有方法，就相当于声明我能够完成某项工作。

实现接口的类继承了定义在接口中的常量，这些类可以使用简单的名字来引用接口定义中的常量。

### 3. 接口与抽象类

根据接口的定义，可以看到接口和抽象类相似，都是只给出方法，没有给出方法具体的实现细节。那么两者是一种怎么样的关系呢？我们可以把接口理解为各个功能模块之间进行联系的协议，为了保证整个系统中各个功能模块的联系，系统就需要定义一系列的接口，这些接口由系统中的功能模块来实现。接口就如同功能模块之间通信的一种规范，一个功能模块可以实现多个接口。

例如，在计算机网络中，可以定义三个网络接口，分别是 RJ45 接口（双绞线）、AUI 接口（粗同轴电缆）和 BNC 接口（细同轴电缆）。我们可以定义网卡和网络设备（如 Hub、Switch、Router）类，来实现上述的一个或多个接口，这样计算机、Hub、Switch 和 Router 就可以连接到网络了。上述问题，用抽象类是不合适的。

**【例 6-6】**　一个有关抽象类和接口的应用实例。

通过下面的例子，演示抽象类、接口、接口的实现的定义和应用，文档名为 HelloWorld.java，内容如下：

```
public class HelloWorld
{
  public static void main(String[] args)
  {
    Dog  animal1 = new Dog();
    Cat  animal2 = new Cat();
    Duck animal3 = new Duck();
    System.out.println("A dog says " + animal1.getHello()
```

```
                                + ", when scared says: " + animal1.getHello(Animal.SCARED)
                                + ", is carnivorous: " + animal1.isCarnivorous()
                                + ", is a mammal: " + animal1.isAMammal());
    System.out.println("A cat says " + animal2.getHello()
                                + ", when comforted says: " + animal2.getHello(Animal.COMFORTED)
                                + ", is carnivorous: " + animal2.isCarnivorous()
                                + ", is a mammal: " + animal2.isAMammal());
    System.out.println("A duck says " + animal3.getHello()
                                + ", when scared says: " + animal3.getHello(Animal.SCARED)
                                + ", is carnivorous: " + animal3.isCarnivorous()
                                + ", is a mammal: " + animal3.isAMammal());
  }
}
// (1)定义抽象类 Animal
abstract class Animal
{
  public static final int SCARED = 1;
  public static final int COMFORTED = 2;
  public boolean isAMammal()        //哺乳动物
  {
    return(true);
  }
  public boolean isCarnivorous()    //食肉类的
  {
    return(true);
  }
  abstract public String getHello();
  abstract public String getHello(int mood);
}
// (2)定义接口 LandAnimal
interface LandAnimal
{
  public int getNumberOfLegs();
  public boolean hasATail();
}
// (3)定义接口 WaterAnimal
interface WaterAnimal
{
  public boolean hasGills();
  public boolean laysEggs();
}
// (4)定义派生类 Dog,实现接口
class Dog extends Animal implements LandAnimal
{
  // 重载父类的方法
  public String getHello()
  {
    return("Bark");
  }
  public String getHello(int mood)
  {
```

```
    switch (mood) {
      case SCARED:
           return("Growl");
      case COMFORTED:
           return("");
    }
    return("Bark");
  }
  // LandAnimal 接口的实现
  public int getNumberOfLegs()
  {
    return(4);
   }
  public boolean hasATail()
  {
    return(true);
  }
}
// (5)定义派生类 Cat,实现接口
class Cat extends Animal implements LandAnimal
{
  // 重载父类的方法
  public String getHello()
  {
    return("Meow");
  }
  public String getHello(int mood)
  {
    switch (mood) {
        case SCARED:
             return("Hiss");
        case COMFORTED:
             return("Purr");
    }
    return("Meow");
  }
  // LandAnimal 接口实现
  public int getNumberOfLegs()
  {
    return(4);
  }
  public boolean hasATail()
  {
    return(true);
  }
}
// (6)定义派生类 Duck,实现接口
class Duck extends Animal implements LandAnimal, WaterAnimal
{
    // 重载父类的方法
    public String getHello()
    {
      return("Quack");
    }
```

```
        public String getHello(int mood)
        {
           switch (mood) {
                case SCARED:
                     return("Quack, Quack, Quack");
                case COMFORTED:
                     return("");
           }
           return("Quack");
         }
        public boolean isAMammal()
        {
           return(false);
        }
        public boolean isCarnivorous()
        {
           return(false);
        }
        // WaterAnimal 接口实现
        public boolean hasGills()
        {
           return(false);
        }
        public boolean laysEggs()
        {
           return(true);
        }
        // LandAnimal 接口实现
        public int getNumberOfLegs()
        {
           return(2);
        }
        public boolean hasATail()
        {
           return(false);
        }
    }
```

执行 javac HelloWorld.java 编译，生成相应的.class 文件。然后执行 java HelloWorld 运行该程序，输出结果如下：

```
A dog says Bark, when scared says: Growl, is carnivorous: true, is a mammal: true
A cat says Meow, when comforted says: Purr, is carnivorous: true, is a mammal: true
A duck says Quack, when scared says: Quack, Quack, Quack, is carnivorous: false, is a mammal: false
```

### 6.2.5　包

在 Java 中，为了管理大型名字空间，避免名字冲突，引入包(Package)的概念，包由一组类和接口构成。一般情况下，每一个类或接口都被存储在不同的文件中，为了管理和使用方便，对于那些相关的类和接口可以绑定到一个包中。例如，Java 的基本类在 java.lang 中，而用于输入和输出的类则在 java.io 中。

Java 中的包其实指的就是目录，它可以更好地管理 Java 类(Class)和接口(Interface)。使用包就定义了一个类和接口的名字空间，一个包中的类和接口的名字与其他包中的名字不会冲突。同时，还可以约束包内的类和外部的类的访问权限。

### 1. 定义包

使用 package 语句可以将一个编译单元(源程序文件)定义成包。如果使用 package 语句，编译单元的第一行必须无空格，也无注释，格式如下：

```
package <包名>;
```

若编译单元无 package 语句，则该单元被置于一个默认的无名的包中。

按照一般的习惯，包名是由“.”号分隔的单词构成，第一个单词通常是开发这个包的组织的名称。

例如，有一个 Java 文件，文件名为 B.java，内容如下：

```
package hao.yy;
public class B {
     B(int yy, int mm, int dd)
     {
       System.out.println("Year:" + yy + "Month:" + mm + "Date" + dd);
     }
}
```

执行 javac -d d:\B.java 命令，其中参数-d <目录>为指定输出文件的目录。因此，上述编译命令执行后将在 d:\目录下创建一个 d:\hao\yy 文件夹，该文件夹中包含了编译后的类文件 B.class。

现在这个包已经创建好了，要使用这个包中的类 B，需要导入，或者把 d:\hao\yy 设置在环境变量 classpath 里。

### 2. 使用包中的类和接口

要使用定义在一个包中的类和接口，可以通过 import 关键词，主要有两种形式：

(1) 使用 import 语句，导入类或接口，或包含它们的包。导入的类和接口的名字在当前的名字空间可用。导入一个包时，则该包中的所有的公有类和接口均可用，形式如下：

```
import java.util.*;
```

其中，第一个语句表示 java.util 中所有的 public 类被导入当前包，“*”表示导入包中的所有类。如果仅仅使用 util 包中的 Date 类，可以写作 import java.util.Date。

例如：

```
import java.util.Date;
import java.text.SimpleDateFormat;
public class mydate {
public static void main(String[] args) {
// Create a date formatter that can parse dates of the form MM-dd-yyyy.
SimpleDateFormat bartDateFormat = new SimpleDateFormat("MM-dd-yyyy");
// Create a string containing a text date to be parsed.
```

```
String dd = "6 - 22 - 2008";
Date date = bartDateFormat.parse(dd);
// Send the parsed date as a long value to the system output.
System.out.println(date.getTime());
}
}
```

(2) 在每个引用的类和接口前面给出它们所在的包的名字,一般形式是:

```
包名.类名 obj = new 包名.<类名|接口名>();
```

例如,要使用上面定义的包 hao.yy 中包含的类,语句为:

```
hao.yy MyTT = new hao.yy.B(2003,11,24);
```

在使用一个外部类或接口时,要声明该类或接口所在的包,否则会产生编译错误。此外,要确保这些类和包的路径正确,即需要包含在 classpath 环境变量中,否则编译器在编译时将找不到所需要的类。

### 6.2.6 Java基础类库

对于所有的程序设计语言,都可以分成两个部分:第一个部分就是语言本身,包括语言的字符集、数据、类型、程序语句、函数等语法部分;第二部分则是开发程序用的标准库,里面包含了大量的标准函数或标准类,是软件开发的 API(Application Program Interface),它可以帮助开发者方便、快捷地开发相应语言的程序。

在 Java 编程中,Java 语言提供了大量的已经实现的标准类,这些类的集合构成 Java 的类库。这些类根据实现的功能不同,划分为不同的集合,每个集合组成一个包,称为类库。Java 类库中大部分都是由 Sun 公司提供的,这些类库称为 Java 基础类库。了解类库的结构可以帮助开发者节省大量的编程时间,而且能够使编写的程序更简单更实用。Java 中丰富的类库资源也是 Java 语言的一大特色,是 Java 程序设计的基础。

#### 1. java.lang 包

java.lang 包又称 Java 语言包,主要含有与语言相关的类。定义了 Java 中的大部分基本类,包含 Java 语言类、线程、异常、系统、Object 类以及各种数据类型等相关的类。java.lang 包是 Java 程序中默认加载的一个包,由解释程序自动加载,不需要显式说明。

掌握类是掌握 Java 程序设计语言的基础。但是,Java 的类库日益庞大,所包含的类和接口众多,用户是无法全部掌握的。下面将简要介绍 Java 类库中的核心部分,即那些最重要、最常用的类和接口,包括 Object、Class、ClassLoader、System、Runtime、Process、String、Collection 等。

(1) 基本类

- java.lang.Object

Object 类是 Java 整个类层次结构的根结点,每个类都使用 Object 作为超类。所有对象(包括数组)都实现这个类的方法。在不明确给出超类的情况下,Java 会自动把 Object 作为要定义类的超类。可以使用类型为 Object 的变量指向任意类型的对象。

Object 类的变量只能用作各种值的通用持有者，要对它们进行任何专门的操作，都需要知道它们的原始类型并进行类型转换。

例如：

```
Object obj = new MyObject();
MyObject x = (MyObject)obj;
```

Object 类有许多方法，关于这些方法的使用请读者参考其他专门讲解 Java 编程的书籍，或者参考 JDK 大全类的技术手册。

• java. lang. Class

Java 程序在运行时，Java 运行时系统一直对所有的对象进行所谓的运行时类型标识，这项信息记录了每个对象所属的类。虚拟机通常使用运行时类型信息来选择正确方法去执行，即实现面向对象中的运行时多态，用来保存这些类型信息的类，这就是 Class 类。Class 类封装一个对象和接口运行时的状态，该类的实例对象表达 Java 应用中正在运行的类或者接口。

Class 类没有公共构造方法，不能由 Java 虚拟机自动实例化，因此不能显式地声明一个 Class 对象。当装载类时，Java 虚拟机以及通过调用类加载器 ClassLoader 中的 defineClass 方法将自动创建 Class 类型。虚拟机为每种类型管理一个独一无二的 Class 对象，也就是说，每个类(型)都有一个 Class 对象。运行程序时，Java 虚拟机(JVM)首先检查所要加载的类对应的 Class 对象是否已经加载。如果没有加载，JVM 就会根据类名查找. class 文件，并将其 Class 对象载入。

注意，基本的 Java 类型(boolean、byte、char、short、int、long、float 和 double)和关键字 void 也都对应一个 Class 对象。每个数组属于被映射为 Class 对象的一个类，所有具有相同元素类型和维数的数组都共享该 Class 对象。一般某个类的 Class 对象被载入内存，它就用来创建这个类的所有对象。

用户不能显式地声明一个 Class 对象，但可以通过调用 Object 类的 getClass()方法来得到 Class 对象，这是最常见的产生 Class 对象的方法。例如：

```
MyObject x;
Class c1 = x.getClass();
```

使用 Class 类中的静态 forName()方法获得与字符串对应的 Class 对象。例如：

```
Class c2 = Class.forName("MyObject");
```

获取 Class 类型对象的第三个方法非常简单。如果 T 是一个 Java 类型，那么 T. class 就代表了匹配的类对象。例如：

```
Class cl1 = Manager.class;
Class cl2 = int.class;
Class cl3 = Double[].class;
```

• java. lang. ClassLoader

该类是 Java 类加载器，负责根据指定的二进制名称加载相应的类。不同的类加载器根据类的二进制名从不同的源中读取二进制的类“ * . class”信息，并生成 Class 对象。每个类

中都有对其加载器的引用。

- java. lang. Syatem

System 类是一个抽象类,所有的字段和方法都是静态的。其中包含一些有用的类字段和方法,它不能被实例化。在 System 类提供的设施中,有三个静态的变量 in、out、err,分别对应标准输入、标准输出和错误输出流;有对外部定义的属性和环境变量的访问的方法、加载文件和库的方法,还有快速复制数组的一部分的实用方法。

因此,System. in、System. out、System. err 实际上表示三个对象,这也就是为什么可以用 System. out. println("Hello World!")的原因。

- java. lang. Runtime

Runtime 类封装了运行时的环境。每个 Java 应用程序都有一个 Runtime 类实例,使应用程序能够与其运行的环境相连接。一般不能实例化一个 Runtime 对象,应用程序也不能创建自己的 Runtime 类实例,但可以通过 Runtime. getRuntime()方法获取当前 Runtime 运行时对象的引用。一旦得到了一个当前的 Runtime 对象的引用,就可以调用 Runtime 对象的方法去控制 Java 虚拟机的状态和行为。

在 Java 语言中,有 8 种基本数据类型,分别是 boolean、byte、char、int、short、long、float 和 double。对应上述 Java 基本数据类型,在 java. lang 包中,定义了一组将原始数据类型对象化的类,分别是 java. lang. Boolean、java. lang. Byte、java. lang. Character、java. lang. Interger、java. lang. Short、java. lang. Float、java. lang. Long、java. lang. Double 等。另外,还定义了一个数字类的父类 java. lang. Number。

所谓原始数据类型对象化就是用类对原始的数据类型进行封装,封装后的类提供了一组对象的操作方法。例如:

Integer 类把 int 基本数据类型包装起来了,提供了一些方法如 parseInt()等。

```
Integer  id = new Integer(x);
Integer  getid()
{
  return new Integer(id.intValue() + 1);
}
```

如果将 id 说明为基本数据类型 int,则类似的功能可以写为 id++。

在 java. lang 包中,还定义了 Math 类和字符串类 String。其中,Math 类提供了常用的数学运算方法以及 Math. PI 和 Math. E 两个数学常量。该类是 final 的,不能被继承,类中的方法和属性全部是静态的,不允许在类的外部创建 Math 类的对象。因此,只能使用 Math 类的方法而不能对其作任何更改。

**【例 6-7】** Math 类应用举例。

下列例子演示了 Math 类的主要属性和方法,代码清单 exa6-7. java 如下:

```
class exa6 {
  public static void main(String args[]) {
    System.out.println("Pi = " + Math.PI);
    System.out.println("E = " + Math.E);
    System.out.println("abs( - 6.8) = " + Math.abs( - 6.8));
    // ceil(d)输出不小于 d 的最小整数(返回值为 double 型)
```

```
        System.out.println("ceil(6.8) = " + Math.ceil(6.8));
        // floor(d)输出不大于 d 的最大整数(返回值为 double 型)
        System.out.println("floor(8.6) = " + Math.floor(8.6));
        System.out.println("round(8.6) = " + Math.round(8.6));
        System.out.println("sqrt(16) = " + Math.sqrt(16));
        System.out.println("exp(1) = " + Math.exp(1));
        System.out.println("log(e) = " + Math.log(Math.E));
        System.out.println("pow(2,3) = " + Math.pow(2,3));
        System.out.println("sin(30degree) = " + Math.sin(Math.toRadians(30)));
        System.out.println("atan(90degree) = " + Math.atan(Math.PI/2));
    }
}
```

编译并运行上述程序,将看到常用的一些数学运算结果。

(2) 基本接口

在 Java 语言包 ava.lang 中定义了一组接口,下面是几个常用的接口:

- java.lang.Appendable:可追加(Appendable)接口。实现该接口的类的对象实例具有可向其追加字符或字符序列的能力。希望能够接收 Formatter 输出的类必须实现该接口。
- java.lang.Cloneable:可克隆(Cloneable)接口。实现了该接口的类具有克隆的能力。可以通过 Object.clone()方法将类的实例对象的域(field)逐个复制到同一个类的另外一个实例中。如果使用 Object.clone()方法克隆没有实现该接口的类的实例对象,将会触发 CloneNotSupportedException 异常。
- java.io.Comparable:可比较接口。实现了该接口的类的两个实例对象之间可以进行比较。比较结果负数(−1)、0、正数(1)分别代表比较对象与被比较对象之间的关系分别是小于、等于、大于。可对实现了该接口的类的多个实例对象进行排序。

#### 2. java.util 包

java.util 包,又称 Java 实用程序包。Java 平台中有两个最常用的基础包,一个是 java.lang 包,另一个就是 java.util 包。java.util 包包含了大量的公用类,包括常用的数学运算类、字符串类、日期、日历类以及向量哈希表等类,还包括一些接口和异常类。

下面重点介绍 Java 中有关日期的类及其操作。日期在 Java 中是非常复杂的,在不同的语言国别环境中,日期的国际化、日期和时间之间的转换、日期的加减运算、日期的展示格式都是非常复杂的问题。

在 Java 中,操作日期主要涉及到以下几个类:

(1) java.util.Date

Date 类封装了有关日期和时间的数据,可以精确到毫秒。从 JDK 1.1 开始,原有的 Date 类中的许多方法已经过时,例如,getYear()、getMonth()、getDay()等,Date 中的把日期解释为年、月、日、小时、分钟和秒值的方法已废弃。现在应该使用 Calendar 类实现日期和时间之间的转换,使用 DateFormat 类来格式化和分析日期字符串。

在 Date 类中,原有的 6 种构造函数中,大部分已经不再使用,主要使用的构造函数是

public Date()和 Date(long date)。其中,Date()用于获取当前的日期和时间,它是程序获得当前系统时间的唯一方法;Date(long date)用于分配 Date 对象并初始化此对象,以表示自从标准基准时间(称为“历元(epoch)”,即 1970 年 1 月 1 日 00:00:00 GMT)以来的指定毫秒数。

例如:

```
Date nowdaytime = new Date(); //获得当前系统的日期时间
System.out.println("Current time is:" + nowdaytime);
```

则输出结果为:Current time is:Sun Jun 22 10:22:27 CST 2008。

(2) java.text.DateFormat

DateFormat 是日期/时间格式化子类的抽象类,它以与语言无关的方式格式化并分析日期或时间。日期/时间格式化子类(如 SimpleDateFormat)允许进行格式化(即日期→文本)、分析(文本→日期)和标准化。将日期表示为 Date 对象,或者表示为从 GMT(格林威治标准时间)1970 年 1 月 1 日 00:00:00 这一刻开始的毫秒数。

(3) java.text.SimpleDateFormat

DateFormat 的直接子类,SimpleDateFormat 是一个以与语言环境相关的方式来格式化和分析日期的具体类。它允许进行格式化(日期→文本)、分析(文本→日期)和规范化。

SimpleDateFormat 使得可以选择任何用户定义的日期-时间格式的模式。但是,仍然建议通过 DateFormat 中的 getTimeInstance、getDateInstance 或 getDateTimeInstance 来创建日期-时间格式化程序。

(4) java.util.Calendar

Calendar 类是一个抽象类,它为特定瞬间与一组诸如 YEAR、MONTH、DAY_OF_MONTH、HOUR 等日历字段之间的转换提供了一些方法,并为操作日历字段(例如获得下星期的日期)提供了一些方法。瞬间可用毫秒值来表示,它是距历元(即格林威治标准时间 1970 年 1 月 1 日的 00:00:00.000,格里高利历)的偏移量。

与其他语言环境敏感类一样,Calendar 提供了一个类方法 getInstance,以获得此类型的一个通用的对象。Calendar 的 getInstance 方法返回一个 Calendar 对象,其日历字段已由当前日期和时间初始化。

(5) java.util.GregorianCalendar

GregorianCalendar 是 Calendar 的一个具体子类,提供了世界上大多数国家使用的标准日历系统。GregorianCalendar 是一种混合日历,在单一间断性的支持下同时支持儒略历和格里高利历系统,在默认情况下,它对应格里高利日历创立时的格里高利历日期(某些国家是在 1582 年 10 月 15 日创立,在其他国家要晚一些)。可由调用方通过调用 setGregorianChange()来更改起始日期。

日期型数据的操作非常复杂,涉及 5 个方面,包括:(1)创建日期;(2)日期格式化显示;(3)日期的转换(主要是和字符串之间的相互转换);(4)日期中年、月、日、时、分、秒、星期、月份等的获取;(5)日期的大小比较、日期的加减。下面是一个关于日期数据操作的综合举例。

【例 6-8】 日期类 Date 应用举例。

```
// TestSimpleDateFormat.java
import java.util.Date;
import java.util.Locale;
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
public class TestSimpleDateFormat {
  public static void main(String args[]) throws ParseException {
    TestSimpleDateFormat test = new TestSimpleDateFormat();
    test.testDateFormat();
  }
  public void testDateFormat() throws ParseException {
    // 创建日期
    Date date = new Date();
    // 创建不同的日期格式
    DateFormat df1 = DateFormat.getInstance();
    DateFormat df2 = DateFormat.getDateInstance(DateFormat.FULL, Locale.CHINA);
    DateFormat df3 = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss EE");
    // 产生一个指定国家指定长度的日期格式,长度不同,显示的日期完整性也不同
    DateFormat df4 = new SimpleDateFormat("yyyy 年 MM 月 dd 日 hh 时 mm 分 ss 秒 EE", Locale.
CHINA);
    DateFormat df5 = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss EEEEEE", Locale.US);
    DateFormat df6 = new SimpleDateFormat("yyyy-MM-dd");
    DateFormat df7 = new SimpleDateFormat("yyyy 年 MM 月 dd 日");
    //将日期按照不同格式进行输出
    System.out.println("(1) 按照默认日期格式,系统默认的区域?:" + df1.format(date));
    System.out.println("(2) 按照指定日期的 FULL 模式,区域设置为中文:" + df2.format
(date));
    System.out.println("(3) 按照指定日期格式 yyyy-MM-dd hh:mm:ss EE,系统默认区域:" +
df3.format(date));
    System.out.println("(4) 按照指定日期格式 yyyy 年 MM 月 dd 日 hh 时 mm 分 ss 秒 EE,区域为中
文:" + df4.format(date));
    System.out.println("(5) 按照指定日期格式 yyyy-MM-dd hh:mm:ss EE,区域为美国:" +
df5.format(date));
    System.out.println("(6) 按照指定日期格式 yyyy-MM-dd,系统默认区域:" + df6.format
(date));
    // 将特定格式的字符串转换为日期,若格式不相配,则会出错
    Date date1 = df1.parse("03-05-11 下午 6:00");
    Date date2 = df2.parse("2003 年 11 月 24 日 星期一");
    Date date3 = df3.parse("2006-5-8 18:30:00 星期一");
    Date date4 = df4.parse("2008 年 6 月 16 日 11 时 30 分 00 秒 星期一");
    Date date5 = df5.parse("2008-06-22 18:30:00 Sunday");
    Date date6 = df6.parse("2008-06-24");
    System.out.println(" ===== 输出不同格式的日期 ====== ");
    System.out.println(date1);
    System.out.println(date2);
    System.out.println(date3);
    System.out.println(date4);
    System.out.println(date5);
    System.out.println(date6);
  }
}
```

实用程序包 java. util 包含的类和接口很多,由于篇幅所限,在此不再介绍,请读者参考有关 Java 编程的专门书籍或 JDK 大全。

### 3. java.awt 包

Java 抽象窗口工具包 java. awt(Abstract Windowing Toolkit)包含一些 GUI 界面相关的类,包括窗口、对话框、菜单、各种控件等。awt 类库还包含一组用来处理绘图、打印和图像,并且支持易用性、拖放和二维图形的 API。通过这些元素,编程者可以控制所写的 Applet 或 Application 的外观界面。

通过 awt 可以创建与平台无关、基于图形用户界面的程序。同微软的 Windows API 相比,清楚、简单和强大的 awt 是 Java 语言迅速流行的重要原因。awt 不仅是编写 Windows 程序的良好工具,而且可以编写其他操作系统平台的图形界面应用程序。

### 4. java.swing 包

新的图形界面类库 java. swing 继承 awt,提供了多种图形界面组件,Swing 中包含了像标签页、表格、树、特殊边框、微调等各种新组件。这些组件都是 100%纯 Java 的,不依赖具体的 Windows 系统,可以在各种平台上实现。Swing 中支持可插入观感(Pluggable look and feel ,PL&F),支持用户定制桌面,更换新的颜色方案,让窗口系统适应特定的用户习惯和需要。Swing PL&F 体系结构使得同时定制 Swing 控件或控件组更加容易。

### 5. java.io 包

输入输出包 java. io,提供了全面的 I/O 接口,包括文件读写、标准设备输出等。它是以流为基础进行输入输出的,所有数据被串行化写入输出流,或者从输入流读入。I/O 体系分为 Input/Output 和 Reader/Writer 两类,区别在于 Reader/Writer 在读写文本时能自动转换内码。流 I/O 的好处是简单易用,缺点是效率较低。

Java 也对块传输提供支持,在核心库 java. nio 中采用的便是块 I/O,块 I/O 效率很高,但编程比较复杂。

### 6. java.applet 包

java. applet 包中定义了设计小应用程序(Applet)的类和接口,包括控制 HTML 文档格式、应用程序中的声音等资源的类。例如 Applet 类、AppletContext 接口、AppletStub 接口、AudioClip 接口等,其中 Applet 是用来创建包含于 HTML 的 Applet 必不可少的。

### 7. java.beans 包

JavaBeans 是 Java 应用程序环境的中性平台组件结构。java. beans 包定义了应用程序编程接口(API),包含与开发 JavaBeans 有关的类和接口。

### 8. java.net 包

含有与网络操作相关的类,如 TCP Sockets、URL 等工具。该包支持 TCP/IP 协议,并包含 Socket 类、URL、与 URL 相关的类。

# 6.3　JavaApplet

JavaApplet 是指用 Java 编写的能够在 Web 页中运行的小应用程序，通常含有可视化内容，并能够产生特殊的显示效果。在 JavaApplet 中，可以实现图形绘制、字体和颜色控制、动画和声音的插入、人机交互及网络交流等功能。

## 6.3.1　Applet 类

在 java.applet 包中，定义了 Applet 类，它是所有 Applet 应用的基类，所有的 Applet 小应用程序都必须继承该类。例如：

```
import java.applet. * ;
public class myApplet extends Applet
{
…
}
```

Applet 类的基本方法见表 6-4。

表 6-4　Applet 类的基本方法

| 方　　法 | 功　　能 |
|---|---|
| Applet() | Applet 类唯一的构造函数 |
| public final void setStub(AppletStub stub) | 设置 Applet 的 stub。stub 是 Java 和 C 之间转换参数并返回值的代码位，由系统自动设定 |
| public boolean isActive() | 判断一个 Applet 程序是否在活动状态 |
| public URL getDocumentBase() | 获得表示该 Applet 运行的文件目录的对象 |
| public URL getCodeBase | 获得该 Applet 代码的 URL 地址 |
| public String getParameter(String name) | 得到由 name 指定的参数值 |
| public AppletContext getAppletContext() | 返回浏览器或小应用程序观察器 |
| public void resize(int width,int height) | 调整 Applet 运行的窗口尺寸 |
| public void resize(dimension) | 调整 Applet 运行的窗口尺寸 |
| public void showStatus(String msg) | 在浏览器的状态条上显示指定信息 |
| public Image getImage(URL url) | 按 URL 指定的地址装入图像 |
| public Image getImage(URL url,String name) | 按 URL 指定的地址和文件名装入图像 |
| public AudioClip getAudioClip(URL url) | 按 URL 指定的地址获取声音文件 |
| public AudioClip getAudioClip(URL url,String name) | 按 URL 指定的地址和文件名获取声音 |
| public String getAppletInfo() | 返回 Applet 的作者、版本、版权等信息 |
| public String[]getParameterInfo() | 返回描述 Applet 参数的字符串数组 |
| public void play(URL url) | 加载并播放一个 URL 指定的音频剪辑 |
| public void destroy() | 撤销 Applet 及其所占用的资源。若该 Applet 是活动的，应先终止该 Applet 的运行 |

Applet 类还有四个基本方法,以用来控制其运行状态:

(1) init()方法:Applet 程序运行前的初始化工作。当一个 Applet 被系统调用时,系统首先调用该方法。通常可以在该方法中完成从网页向 Applet 传递参数,添加用户界面的基本组件等操作。

(2) start()方法:系统在调用完 init()方法之后,将自动调用 start()方法。而且,每当用户离开包含该 Applet 的 Web 页后再次返回时,系统将再次执行 start()方法。这就意味着 start()方法可以被多次执行,而不像 init()方法。因此,可把只希望执行一次的代码放在 init()方法中。可以在 start()方法中开始一个线程,如继续一个动画、声音等。

(3) stop()方法:当用户离开 Applet 程序所在的网页时,运行该方法。因此它也可以运行多次。

(4) destroy()方法:停止 Applet 程序的运行,撤销 Applet 及其所占用的资源。当在关闭浏览器时执行该方法。

**【例 6-9】** 使用 JavaApplet 举例。

下面是一个创建 JavaApplet 并在网页中使用的例子,以此来说明 JavaApplet 的使用步骤。

(1) 编辑 JavaApplet 源程序,因为 JavaApplet 为 public 类,因此,文档名应和 Applet 类名一致。helloApplet 类定义如下:

```
// 文档名为 helloApplet.java
import java.awt.*;
import java.applet.*;
public class helloApplet extends Applet
{
  public void paint(Graphics g)
  {
     g.drawString("Hello World!",5,35);
  }
}
```

将该文档保存在 d:\myjava 文件夹下,文档名为 helloApplet.java。

(2) 把 Applet 的.java 源程序编译为字节码.class 文件,使用 javac helloApplet.java 命令编译该文件,生成.class 文件。

(3) 制作使用 JavaApplet 的 HTML 文件 myApplet.htm。在 HTML 文件内插入 <applet></applet>语句,引入上述的.class 文件。

创建的 HTML 文档 myApplet.htm 保存在和.class 相同的文件夹中,内容如下:

```
<html>
<title>helloApplet</title>
<applet code = "helloApplet.class" width = 200 height = 100>
</applet>
</html>
```

双击 HTML 文档,在浏览器中将看到 Applet 的执行结果。

### 6.3.2 Applet 交互

JavaApplet 还经常用于用户交互，Applet 支持鼠标、键盘等多种事件，用户可以对 Applet 进行交互控制。

（1）Applet 响应的鼠标事件

JavaApplet 支持的鼠标操作有 mouseUp、mouseDrown、mouseDrag、mouseEnter 和 mouseExit 等。在 Applet 类中包含了三种事件的处理方法，用户只要在各个方法中加入适当的事件处理代码，即可完成相应事件的处理工作。

Applet 类鼠标事件处理方法的首部如下，其中形参 x 和 y 是事件发生时鼠标指针的位置：

- public boolean mouseDown(Event event,int x,int y)：按下鼠标按键。
- public boolean mouseUp(Event event,int x,int y)：放开鼠标按键。
- public boolean mouseDrag(Event event,int x,int y)：拖动鼠标。
- public boolean mouseEnter(Event e,int x,int y)：鼠标指针指在对象上。
- public boolean mouseExit(Event e,int x,int y)：鼠标指针离开对象。

（2）JavaApplet 响应的键盘事件

JavaApplet 目前支持两种键盘事件：KEY_PRESS 和 KEY_RELEASE。当按下键盘的某一键时发生 KEY_PRESS 事件；当放开被按下的键时发生 KEY_RELEASE 事件。Applet 类的 keyDown()方法和 keyUp()方法处理键盘事件，两方法的首部如下：

- public boolean keyDown(Event event,int KeyPressed)
- public boolean keyUp(Event event,int KeyPressed)

形参 event 表示事件对象本身，该对象的 id 成员保存了事件发生时对象的当前值，它与 Event 对象的一组成员变量相对应。例如，通过 event. id 的值是否为成员变量 Event. KEY_ACTION 可知道是否按了功能键。绝大多数键盘操作都有与之对应的 Event 对象成员变量。Event 的成员变量很多，请参考其他相关书籍。

形参 KeyPressed 对应于所按键的值。对于常规键，其值为 ASCII 码，其他键也都有确定的值，如 Esc 键的键值是 27。

**【例 6-10】** 设计一个 JavaApplet 程序，当在 Applet 窗口内单击鼠标时显示鼠标指针的(x,y)坐标值。当按键时显示所按键的键值和对应的字符。

JavaApplet 代码如下，文档名为 MouseAndKeyApplet. java。

```
import java.awt.Event;
import java.awt.Graphics;
import java.applet. * ;
public class MouseAndKeyApplet extends Applet
{
  String mouseDownInf = null;              //保存按下鼠标时的显示信息
  String mouseDragInf = null;              //保存拖动鼠标时的显示信息
  String mouseEnterInf = null;             //保存鼠标指向或离开 Applet 对象时的显示信息
  String keyDownInf = null;                //保存键盘操作信息
  public boolean keyDown(Event evt, int key)
```

```
    { //按下键盘键时发生的事件
      keyDownInf = "按键：" + (char)key + " 键值：" + key;
      repaint();                         //重画对象
      return true;                       //返回 true 表示事件处理成功
    }
    public boolean mouseDown(Event evt, int x, int y)
    { //单击鼠标左键或右键时发生的事件
      mouseDownInf = "单击位置：(" + x + ", " + y + ")";
      repaint();
      return true;
    }
    public boolean mouseDrag(Event evt, int x, int y)
    { //按住鼠标按键拖动时发生的事件
      mouseDragInf = "拖动位置：(" + x + ", " + y + ")";
      repaint();
      return true;
    }
    public boolean mouseEnter(Event evt, int x, int y)
    { //鼠标指针指向 Applet 对象时发生的事件
      mouseEnterInf = "Hi,Welcome!";
      repaint();
      return true;
    }
    public boolean mouseExit(Event evt, int x, int y)
    { // 鼠标指针离开 Applet 对象时发生的事件
      mouseEnterInf = "Left Applet";
      repaint();
      return true;
    }
    public void paint(Graphics g)
    { // 显示 Applet 对象的方法
      if (mouseDownInf! = null)
         g.drawString(mouseDownInf,25,90);
      if (mouseDragInf! = null)
         g.drawString(mouseDragInf,150,90);
      if (keyDownInf! = null)
         g.drawString(keyDownInf,280,90);
      if(mouseEnterInf! = null)
         g.drawString(mouseEnterInf,200,45);
    }
}
```

下面是调用 JavaApplet 的 HTML 文档，文档名为 myapplet2.htm，内容如下：

```
<html>
<head>
</head>
<body>
<p><font color = "#FF0000" size = "5">JavaApplet 中的鼠标和键盘事件演示</font></p>
<hr>
<applet code  =  "MouseAndKeyApplet.class" width = 500 height = 100>
```

```
</applet>
</body>
</html>
```

在浏览器中打开 myapplet2.htm 文档，显示结果如图 6-6 所示。




图 6-6　JavaApplet 在浏览器中的显示

最后需要强调的是，JavaApplet 并不是一个应用程序，它没有一个包含 main()方法的类。它只是一个由已在运行的 Java 应用程序（如 Web 浏览器或 applet 查看器）装入并运行的 Java 类。

## 6.3.3　在 HTML 中使用 JavaApplet

JavaApplet 的主要应用就是插入到 HTML 页面中，以产生特殊的显示效果。可以用 JavaApplet 实现文字、图片等特效，例如一种类似湖面效果的特效。还可以使用 JavaApplet 进行人机交互，比如一张图片，当鼠标移到上面会产生波动，就像现实中手波及水面使得水开始波动。

要在 HTML 页面中插入 JavaApplet，首先编写 Apple 源程序，扩展名为.java。编辑完成后使用 javac 命令将它编译转换成字节码文件(.class 文件)，然后再在 HTML 文档中通过加入<applet></applet>标记的方式将 Java 小程序引入到 HTML 文档中。

在 HTML 中使用 JavaApplet 的一般形式是：

```
<applet width = " " height = " " code = "myapplet.class" codebase = " ">
      浏览器不支持 Java 的提示信息
</applet>
```

JavaApplet 的可执行代码为.class 文件，JavaApplet 具有安全、功能强和跨平台等特性，所有主流的浏览器都能够显示包含 Applet 的页面。当用户访问包含 JavaApplet 的网页时，Applet 被下载到用户的本地计算机上执行，因此执行速度不受网络带宽的限制，用户可以更好地欣赏网页上 Applet 产生的多媒体效果。

通过<applet>标记来定义 Applet 程序的使用，<applet>标记所包含的属性见表 6-5。

**表 6-5 Applet 标记常用属性**

| 属 性 | 作 用 |
|---|---|
| code | Applet 的 class 文件名 |
| codebase | 指定 Applet(.class 文件)所在的 URL 地址。它可以是绝对地址，如 www.sun.com。也可以是相对于当前 HTML 文档所在目录的相对地址，如/AppletPath/Name。如果不指定 codebase 属性，浏览器将使用和 HTML 文件相同的 URL |
| alt | 浏览器不支持 JavaApplet 时显示的信息 |
| width、height | Applet 窗口的大小 |
| align | Applet 窗口的显示位置，取值可以是 top、middle、bottom |
| vspace、hspace | 在 Applet 窗口周围的水平和垂直空白条的尺寸 |
| name | 把指定的名字赋给 Applet 的当前实例 |
| param | 指定 JvavaApplet 参数。格式为：PARAM Name="name" VALUE="Liter" |

最后需要说明的是，JavaApplet 是从远程服务器上下载而在本地机上运行的，由于安全方面的原因，浏览器通常对它的运行进行了必要的限制。例如，永远无法运行本地机上的程序；只能与它所在的服务器联系；无法对本地机上的文件进行读写操作；除了本地机使用的 Java 版本号、操作系统名称及版本号文件名分隔符、文件路径外，无法获得本地机的其他信息。此外，由于 Flash 的出现，JavaApplet 的功能也可以用 Flash 来实现，因此，JavaApplet 的应用已经大不如从前了。

## 6.4 JavaBeans

JavaBeans 组件是一种开发软件组件的技术，软件组件技术可以把软件功能作为一个组件来组装成一个应用程序。在 Web 应用开发中，JavaBeans 通常用作中间“商业逻辑”层，用于将表示逻辑和数据访问逻辑隔离，许多业务逻辑通常被编码成 JavaBeans，而不是直接写在 JSP 页面中。由于 CGI/Servlet 的复杂性，目前 JavaBeans 和 JSP 技术已经成为 Web 应用服务器三层结构中的中间层业务逻辑开发的主要工具。

### 6.4.1 什么是 JavaBeans

什么是 JavaBeans 呢？简单地讲，JavaBeans 是符合一定规范的 Java 类。任何具有某种特性和事件接口约定的 Java 类都可以写成一个 Bean。通过将一个满足某些准则的 Java 类写成 JavaBeans，可以有效地实施软件的重用。

具体地讲，JavaBeans 有两层含义：首先，JavaBeans 是一种规范，一种在 Java(包括 JSP)中可重复使用的 Java 组件的技术规范；其次，JavaBeans 是一个 Java 的类，一般来说，这样的 Java 类将对应于一个独立的.java 文件，在绝大多数情况下，它是一个 public 类型的类。

从原理上来说，使用 JavaBeans 就是按照一种特定的规范在 Java 程序中引入一个类。

对于一个简单的 Java 应用程序，完全可以不必拘泥于使用 JavaBeans。即便是对于 JSP，也可以做同样的选择，因为在“<%”和“%>”中可以使用任何纯 Java 语句，包括对一个新对象的定义和实例化。

但是，在绝大多数情况下，使用 JavaBeans 是一个明智的选择。因为使用 JavaBeans 可以充分利用组件的可重复使用的特性并增加程序的可读性。当应用系统日趋庞大时，遵守 JavaBeans 规范将给系统开发和维护带来极大的便利，使得开发 JavaBeans 所受到的约束变得微不足道。

## 6.4.2 JavaBean 的属性、方法和事件

一个 JavaBean 和一个 JavaApplet 类似，是一个遵循某些规则的 Java 类，每个 JavaBean 的功能可能不同，但一般都遵循以下特征：

(1) 支持自检。这样构造器(builder)可以分析 bean 是如何动作的。

(2) 支持定制。用户可以通过应用程序构造器工具定制 bean 的外观和行为。

(3) 支持事件。保证 bean 和外部的通信。

(4) 支持属性。使得 bean 真正地具有内部状态，从而根据应用需要定制。

(5) 支持持久性。这样 bean 才能在应用程序构造器工具中定制，并将定制的状态存储起来以便以后使用。

### 1. 方法

JavaBean 归根到底是一个 Java 类，所有的 public 方法都可以通过对象外部直接调用。JavaBean 中常用的方法是单值属性设置器/获取器，即 set 属性值和 get 属性值。

### 2. 属性

JavaBean 提供了高层的属性(properties)概念。从面向对象的角度看，属性就是传统的对象属性(attribute)。JavaBean 属性描述了 bean 的内部状态，是 JavaBean 中的数据部分，属性的值可以通过适当的 bean 方法进行读写。

JavaBean 有四种类型的属性，分别是：单值属性、索引属性、关联属性和限制属性。

(1) 单值属性

单值属性是 JavaBean 中最简单的属性，只需要定义一个包含一个值的数据成员，并为其定义一对设置(set)/获取(get)属性的方法，以便于外部与其发生联系。如果没有为单值属性提供设置器方法，则该属性为只读型属性；如果没有为单值属性提供获取器方法，则该属性为只写属性。

单值属性设置器/获取器定义的一般形式是：

```
public void set<PropertyName>(<PropertyType> propertyValue)   // 设置器
public <PropertyType> get<PropertyName>()                     // 获取器
```

(2) 索引属性

索引属性类似于 Java 中的数组，包括若干个数据类型相同的元素，可以通过整数索引

值访问其中的属性,因此称为索引属性。

(3) 关联属性

JavaBean API 除了支持单值属性和索引属性外,还提供了一些属性用于增强 JavaBean 的属性管理功能。如关联属性,当修改这类属性时,将发送一个通知给其他元素(如 Applet、application 或其他 JavaBean),如果与该 JavaBean 中的某个属性相关联,就会注册该属性。因此,只要这个关联属性发生变化,就会有一个通知发送给这些相关部件,这些属性称为关联属性,它们的值发生改变与外部部件有关,外部部件称为监听器。

一个有关联属性的 JavaBean 需要支持如下一对事件监听器的注册方法:

```
public void addPropertyChangeListner(PropertyChangeListner1)
public void removePropertyChangeListner(PropertyChangeListner1)
```

(4) 限制属性

JavaBean API 中的另一种高级属性类型是限制(constrained)属性,它可以使外部部件在接受属性的修改值之前先确认修改值。也就是说,当需要修改一个限制属性值时,接受属性的外部部件首先要检查这个属性的合理性再决定是否接受修改。

### 3. 事件

JavaBean 和其他软件组件交流信息的主要手段是发送和接收事件。在新的 AWT 事件模型中,一个事件源可以注册事件监听器对象,当事件源检测到某种事件发生时,将激活检测器对象中一个相应的事件处理方法。

JavaBean 必须能够发送事件变化通知和监听到其他 JavaBean 的事件变化,并对监听到的事件变化进行相应的业务处理。事件的监听原理是:首先事件源必须对需要发送的事件进行注册,然后注册事件监听器,并说明该事件源所发生的事件向什么组件发送,也就是说,在事件源组件中实现方法并在监听组件中注册该事件源。

**【例 6-11】** JavaBean 应用举例。

在 Web 应用中,用户定义的类或 JavaBeans 通常存储在根目录下的"WEB-INF\classes\包\"文件夹中,Java 通过包来实现用户类定义的分类存储和管理。

一个 NameCard 的 JavaBean 代码如下(文件名为 NameCard.java):

```
package cards;
public class NameCard
{
   String Name,Address;
   public NameCard()
   {
     this.Name = "John";
     this.Address = "No,Road,City,Country";
   }
   public void setName(String myName)
   {
     this.Name = myName;
   }
   public String getName()
```

```
        {
          return (this.Name);
        }
        public void setAddress (String myAddress)
        {
          this.Address = myAddress;
        }
        public String getAddress ()
        {
          return (this.Address);
        }
    }
```

上述 JavaBean 应存储在 Web 应用特定的文件夹中，即 Web 应用根目录中"WEB-INF\classes\cards"下，用 javac NameCard.java 编译生成 NameCard.class，该文件将存储在 cards 包中，即 cards 子文件夹中。

当 JavaBean 完成后，为了测试 JavaBean 的功能，可以在 bean 类中临时增加 main 方法，借用 main()方法来调试 bean。JavaBean 的主要应用是将 JavaBean 应用于 JSP 页面中，关于在 JSP 中使用 JavaBean，将在 6.6.4 小节进行详细介绍。

### 6.4.3 Enterprise JavaBeans

1997 年 4 月 12 日，Sun 宣布了一项为企业环境开发 Java 平台的创新成果，即 Java 2 Platform，Enterprise Edition(J2EE)。J2EE 本质上是由若干技术构成的一个 Java 开发框架，这些主要的 J2EE 技术包括：

- Enterprise JavaBeans(EJB)技术
- Java Interface Definition Language(IDL)
- Java Message Service(JMS)API
- Java Naming and Directory Interface(JNDI)
- Java Remote Method Invocation(RMI)和 Object Serialization
- JavaServlet API
- Java Transaction API(JTA)
- Java Transaction Service(JTS)
- JavaServer Pages(JSP)技术
- JDBC 数据库访问 API

虽然从名称上看，Enterprise JavaBeans(EJB)和 JavaBeans 类似，容易令人误认为 JavaBeans 是用于客户端的开发，Enterprise JavaBeans 是用于服务器端的开发，这种来自字面上的理解是完全错误的，因为两者是完全不同的概念。

JavaBeans 和 EJB 都是组件模型规范，但是前者说明了系统开发中业务逻辑的模块化、组件的定义和应用程序组装的问题，而后者则侧重于部署组件的服务框架的细节，即 Enterprise JavaBeans 的侧重点是详细地定义了一个可以部署 Java 组件的服务框架模型。EJB 不涉及属性、发送和接收事件，它定义了一个企业应用的框架模型，具体地说，EJB 是一

个技术规范,它描述了构建应用组件要解决的可扩展(Scalable)、分布式(Distributed)、事务处理(Transactional)、数据存储(Persistent)和安全性(Secure)等问题,以指导软件组件的开发和部署。关于 EJB 的更多内容,请读者参考其他的相关书籍。

## 6.5 Servlet 服务器程序

Java 应用程序可以在服务器上运行,但是不管是在客户机/服务器环境下,还是在基于 Web 的环境下,JDK 中都没有提供让 Java 应用程序专用于服务器机器的接口或包。认识到 Java 在 Web 环境下作为一种服务器语言的潜力,Sun 编写了 JavaServlet 规范。Servlet 在许多方面与 applet 相似,它是专门为在 Web 服务器机器上运行而设计的 Java 程序。

### 6.5.1 Servlet 与 CGI

在一台 Web 服务器控制下,在多台服务器上运行若干小型用户程序,这就是公共网关接口(CGI)程序(又称 CGI 脚本)所起的作用。CGI 应用程序本身往往不是完整的应用程序,在处理接收自 Web 浏览器上用户的信息请求时,CGI 只是整个处理过程中的一个中间步骤。例如,CGI 应用程序的一种常见用途是访问数据库。将它用于这种任务时,CGI 程序提供一种方法,将用户的数据请求连接到能满足这种请求的企业数据库。CGI 程序常常充当一种中间软件,从 Web 浏览器接收请求,决定必须调用哪些计算资源来满足这些请求,并向浏览器发回响应。传统的 CGI 程序一般是由 C 或 C++开发的,这些语言的执行速度较快。随着优化编译器的引入,用 Java 来实现中间层成为可能。

Servlet 是专门为在 Web 服务器机器上运行而设计的 Java 程序,相当于运行于服务端的 applet,与 CGI 程序一样,用于充当连接前端 Web 请求与后端数据资源的中间层组件。它可以动态地扩展服务器的能力,并采用请求-响应模式提供 Web 服务。使用 JavaServlet 可以以更高的效率和可移植性来实现 CGI 的目的,并最终会取代传统的 CGI 程序。

### 6.5.2 Servlet 编程

在现在的 Web 开发中,传统的 CGI 编程正在被 Servlet 技术所替代。JavaServlet 的出现,为应用程序员使用 Java 来创建 Web 应用程序开辟了新的途径。但是,Servlet 只是工作在 Web 服务器上的一个连接客户请求和数据库系统的中间层,它不是一个可以独立运行的 Java 程序,它不能在操作系统下直接运行,必须运行在 Servlet 容器中。

#### 1. 编写 Servlet 所需要的开发环境

在服务端,要运行 Servlet,需要安装支持 JavaServlet 的 Web 服务器。最新的 Web 服务器或应用服务器,例如 Tomcat,已经配置了运行 Servlet 和 JSP 所有必需的软件。因此,Tomcat 既可以单独作为小型 Servlet、JSP 测试服务器,也可以集成到 Apache Web 服务器中,构成一个 Web 应用服务器。

### 2. Servlet的开发过程

Servlet 没有 main() 函数，因此它不是一个真正意义上的完整的独立的 Java 程序。Servlet 只是运行在 Web 服务器上的一个负责用户和数据层之间转换的接口，它是用 Java 编写的类。所有的 Servlet 都必须继承基本的 Servlet 类，定义请求处理的方法。

要开发 Servlet，需要使用 JavaServlet API 扩展程序包 javax. servlet 和 javax. servlet. http。javax. servlet. http 包定义了利用 HTTP 协议与客户端进行交互的类与界面，javax. servlet 包适用于基于客户自定义协议的 Servlet 开发。

下面的代码显示了一个简单的 Servlet 的基本结构。

```
import java.io. * ;
import javax.servlet. * ;
import javax.servlet.http. * ;
public class SomeServlet extends HttpServlet {
      public void doGet(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException
      {
          // 使用"request"读取和请求有关的信息(比如 Cookies)和表单数据
          // 使用"response"指定 HTTP 应答状态代码和应答头
          // (比如指定内容类型,设置 Cookie)
          // 使用 "out"把应答内容发送到浏览器
          PrintWriter out = response.getWriter();
      }
}
```

如果某个类要成为 Servlet，则它必须从 HttpServlet 继承，根据数据判断是通过 GET 还是 POST 发送，覆盖 doGet、doPost 方法之一或全部。doGet 和 doPost 方法都有两个参数，分别为 HttpServletRequest 类型和 HttpServletResponse 类型。HttpServletRequest 提供访问有关请求的信息的方法，例如表单数据、HTTP 请求头等等。HttpServletResponse 除了提供用于指定 HTTP 应答状态、应答头(Content-Type、Set-Cookie 等)的方法之外，最重要的是它提供了一个用于向客户端发送数据的 PrintWriter。对于简单的 Servlet 来说，它的大部分工作是通过 println 语句生成向客户端发送的页面。

**注意**：doGet 和 doPost 抛出两个异常，必须在声明中包含它们。另外，还必须导入 java. io 包(要用到 PrintWriter 等类)、javax. servlet 包(要用到 HttpServlet 等类)以及 javax. servlet. http 包(要用到 HttpServletRequest 类和 HttpServletResponse 类)。

最后，doGet 和 doPost 这两个方法是由 service 方法调用的，有时可能需要直接覆盖 service 方法，比如 Servlet 要处理 GET 和 POST 两种请求时。

**【例 6-12】**　一个简单的 servlet 代码示例(HelloServlet. java)。

```
package test;
import java.io. * ;
import javax.servlet. * ;
import javax.servlet.http. * ;
public class HelloServlet extends HttpServlet
{
```

```
public void doGet(HttpServletRequest request,HttpServletResponse response)
        throws ServletException,IOException
  {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<html><head><title>");
    out.println("This is my first Servlet");
    out.println("</title></head><body>");
    out.println("<center><h1>Hello,Servlet!</h1></center>");
    out.println("</body></html>");
  }
}
```

该 servlet 实现如下功能：当用户通过浏览器访问该 servlet 时，该 servlet 向客户端浏览器返回一个 HTML 页面，显示 Hello Servlet!。

在 Web 开发中，Servlet 通常需要存储在特定的文件夹下，否则在编译时或运行时将出现找不到 Servlet 的错误。在 Web 应用根目录下，创建 WEB-INF 文件夹，包含 class 文件夹，在 class 文件夹下，可以定义另外的子文件夹，分别存储用户 Web 项目中的 Java 类。

(1) 编译 Servlet 代码

当 Servlet 编写完成后，使用 javac HelloServlet.java 命令编译上述的 Servlet 代码。如果在编译过程中提示 javax.servlet 包不存在，通常是当前的环境变量设置有问题，例如 classpath 路径中缺少一个解析 servlet 类的包。

在 tomcat 中，需要将\common\lib 目录下的 servlet-api.jar 添加到 classpath 中。具体方法是：复制 servlet-api.jar 文件，添加到 classpath 中，或复制到 jre 的扩展包路径\jre\lib\ext 中，再次编译，则生成 MyServlet.class 文件，编译成功。

(2) 运行 Servlet

Servlet 不是一个可执行的 Java 程序，它是在 Servlet 容器(如 Tomcat)中运行的。现在对 MyServlet 进行测试，打开浏览器，按照通常的思路，在地址栏内输入：

```
http://127.0.0.1/HelloServlet
```

在 Tomcat 中，运行 Servlet 比较复杂，如果显示错误信息，找不到要运行的 Servlet，则需要再对 Tomcat 进行相应的配置，主要的配置包括：

首先，修改%TOMCAT_HOME%\conf\Web.xml 文件。找到<servlet>元素，内容如下：

```
<!--
<servlet>
        <servlet-name>invoker</servlet-name>
        <servlet-class>
          org.apache.catalina.servlets.InvokerServlet
        </servlet-class>
        <init-param>
            <param-name>debug</param-name>
            <param-value>0</param-value>
        </init-param>
        <load-on-startup>2</load-on-startup>
```

```
</servlet>
-->
```

然后再找到下述代码：

```
<!-- The mapping for the invoker servlet -->
<!--
      <servlet-mapping>
            <servlet-name>invoker</servlet-name>
            <url-pattern>/servlet/*</url-pattern>
      </servlet-mapping>
-->
```

上述代码显示 Tomcat 调用器(InvokerServlet)被关闭了。在 Tomcat4.1.12 版本之前，默认情况下调用器是启用的，由于一个安全缺陷的揭示，因此，在以后的版本中，调用器默认情况下是禁用的。如果要启用的话，只要将%TOMCAT_HOME%\conf\Web.xml 文件中上述两段代码的注释取消即可。

除了通过浏览器地址栏直接运行 Servlet 外，还可以写一个调用该 Servlet 的 HTML 文件，通过 HTML 的 form 表单中的 action 调用 Servlet，运行相应的 Servlet。例如：

```
<html>
<head>
    <title>JavaServlet s Sample</title>
</head>
<body>
    <form method="get" action="test.HelloServlet">
        <input name="test" type="submit" value="Test HelloServlet">
    </form>
</body>
</html>
```

单独编写 Servlet 非常麻烦，随着 JSP 技术的发展，可以直接将 Java 代码写在 HTML 页面中，这些代码被包含在“<%”和“%>”内，形成 JSP 文件。对于 JSP 页面中的代码，将被编译成 Servlet，这样就大大地减少了 Servlet 开发的难度。

## 6.6　JSP 技术

使用 Servlet 开发服务端中间层逻辑，实在是太复杂了。为此，Sun 公司于 1999 年推出 JSP(Java Server Pages)技术。JSP 是一种 Java 平台技术，它是在传统的 HTML 文档(*.htm，*.html)中加入 Java 脚本程序(Scriptlet)和 JSP 标记来构成的，它包装了 JavaServlet 系统界面，用户可以在 JSP 页面上直接书写 Java 代码，这样就大大简化了 Java 和 Servlet 的使用难度。此外，作为一种服务器脚本语言，JSP 是在编译成 Servlet 后才能实际运行。当用户浏览.jsp 网页时，首先在服务器端执行.jsp 文档中的服务端代码，然后把执行结果传送到客户端浏览器，基本上与浏览器无关。

### 6.6.1 JSP 的运行环境

要运行 JSP 页面，在 Web 服务器端，需要安装 Tomcat 应用服务器。Tomcat 通常和 Apache Web 服务器一起共同构成一个 Web 站点。

#### 1. 运行与开发环境

JSP 的运行和开发环境框架模型如图 6-7 所示。




图 6-7 JSP 的运行和开发环境

JSP 的开发、运行环境很多，可以采用 UNIX、Linux、Windows Server 等不同类型的操作系统，如果希望在服务器上开发 Servlet/EJB/JSP 应用程序，还应该安装以下软件：

(1) 安装 Java 环境。

(2) 安装 Java VM(JRE)，Tomcat 需要 Java VM 的支持。

(3) 安装 Tomcat。

Tomcat 是针对于 Apache 服务器开发的 JSP 应用服务器，是 JavaServlet 和 Java Server Pages 技术的标准实现，是基于 Apache 许可证下开发的自由软件。可以这样认为，当在一台机器上配置好 Apache 服务器，可利用它响应对 HTML 页面的访问请求。实际上 Tomcat 部分是 Apache 服务器的扩展，但它是独立运行的，所以当运行 Tomcat 时，它实际上作为一个与 Apache 独立的进程单独运行。当配置正确时，Apache 为.html 页面服务，而 Tomcat 实际上运行.jsp 页面和 Servlet。

(4) Apache 服务器

可以根据需要安装 Apache 服务器。因为在这一章我们主要是介绍 JSP 的开发，因此只要安装了 Tomcat 就可以了。

#### 2. 建立一个 JSP 程序的测试站点

为了测试后面要学习的 JSP 程序，我们需要建立一个 JSP 程序测试站点。步骤如下：

(1) 建立测试站点的主目录

假设测试站点的主目录是 D:\MyJSP，然后所有的 JSP 程序就可以存储到该目录下，或者建立虚拟目录。

在主目录下，创建一个 index.jsp 文件。

(2) 修改 Tomcat 的默认设置

打开 Tomcat 的主配置文件\Tomcat 6.0\conf\server.xml，具体修改包括两个方面：

① 将 Tomcat 默认的 Web 服务端口号 8080 改为 80。

② 增加新 Web 应用的上下文。

例如，＜Context path＝"/myjsp" docBase＝"d:\MyJSP\" debug＝"0" reloadable＝"true" privileged＝"true"/＞。

具体修改见第 2 章 Web 服务器的架设和管理，它定义了一个虚拟目录对应的物理目录，告诉 Tomcat 如何找到用户的 Web 应用。这样，就可以在浏览器地址栏中输入 http://127.0.0.1:8080/myjsp/文件.jsp 访问一个 JSP 文档了。

当上述修改完毕后，在任务栏中右击 Tomcat 图标，选择“Shutdown：Tomcat”命令，关闭 Tomcat。然后在“开始”菜单中重新启动 Tomcat，尝试运行用户 Web 应用。

**注意**：*如果安装了 Apache，在计算机启动时它将被启动，此时需要首先将 Apache Http Server 关闭，然后再运行 Tomcat。*

(3) 运行 JSP 程序

如果要运行 D:\MyJSP 下面的 JSP 文档 1.jsp，在地址栏中输入：http://127.0.0.1/myjsp/1.jsp。注意：必须要输入扩展名，同时要注意文件名和目录的大小写。其中，myjsp 为虚拟目录。

## 6.6.2 JSP 的语法结构

JSP 文档是通过在 HTML 文档中加入 Java 脚本程序构成的。Java 脚本程序代码用“＜%”和“%＞”标记括起来，称为 JSP 元素。JSP 元素可以分为三种类型：脚本元素、指令元素和动作元素。脚本元素规范 JSP 中所使用的 Java 代码；指令元素针对 JSP 引擎控制转译后的 Servlet 的整个结构；动作元素主要连接到用到的组件(如 JavaBeans 和 Plugin)，另外它还可以控制 JSP 引擎的行为。

### 1. JSP 指令

JSP 指令是 JSP 的引擎。它们不直接产生任何可视的输出，只是指示引擎如何处理 JSP 页面中的内容。JSP 指令由＜%@ …%＞标记，一般形式是：

```
<%@ 指令名 属性1="属性值" 属性2="属性值" … 属性n="属性值" %>
```

主要的两种指令是 page 和 include。

(1) page 指令

page 指令用来定义整个页面的相关属性。该指令的语法和它所包含的属性如下：

```
<%@   page
    [ language="java" ]
    [ extends="package.class" ]
    [ import="{package.class | package.*}, …" ]
    [ session="true | false" ]
    [ buffer="none | 8kb ]
    [ autoFlush="true | false" ]
    [ isThreadSafe="true | false" ]
    [ info="text" ]
```

```
  [ errorPage = "relativeURL" ]
  [ contentType = "mimeType [ ;charset = characterSet ]" |
                 "text/html ; charset = ISO - 8859 - 1" ]
  [ isErrorPage = "true | false" ] %>
```

下面是 page 指令的几个常用属性：

- language 属性：所使用的脚本语言。例如，<%@ page language="Java" %>。
- import 属性：脚本元素中使用的类，与 Java 中的 import 声明作用相同，应是类的全名，或者类所在的包。例如：

```
<%@ page import = "java.util.Date" %>
<%@ page import = "java.io. * " %> (java.io 包中的所有类在本页中都可以使用)
```

- session 属性：是否使用 session 对象。
- buffer 属性：对象 out 的输出模式。none 为没有缓冲区；8KB 为缓冲区大小。
- autoFlush 属性：缓冲区已满时是否自动清空。当 buffer 为 none 时该属性不能为 false。
- isThreadSafe 属性：处理对象间的存取是否引入 Thread Safe 机制。如果为 true，则在程序中必须有多线程的程序代码，否则直接实现 SingleThreadModel 机制。
- errorPage 属性：设置异常处理网页。
- IsErrorPage 属性：当前网页是否是另一个 JSP 网页的异常处理网页。
- contentType 属性：输出到客户端的 MIME 类型和字符集。默认的字符集是 ISO-8859-1。例如，使用中文字符集：<%@ page contentType="gb2312" %>。

可以在几乎所有的 JSP 页面顶部找到指令 page，使用 page 指令(at page)可以定义网页的处理方式，如到何处寻找 Java 类支持等。常用的有：

```
<%@ page import = "java.util.Date" %>
<%@ page errorPage = "errorPage.jsp" %>(当出现错误时的错误处理网页)
<%@ page session = "true" %>
```

(2) include 指令

这是一个在 JSP 网页中包含其他文件的指令，它是在 JSP 引擎将它转译成 Servlet 时产生作用的指令，其一般形式如下：

```
<%@ include file = "插入文件的 URL" %>
```

这里要求，被包含进的文件必须符合 JSP 的语法，应是静态文本、指令元素、脚本元素和动作元素。注意，包含后面不能有分号。

### 2. 声明

JSP 的声明用于定义页面级的变量，如果代码太多的话，最好写入一个单独的 Java 类中。声明由<%! … %>定义。必须通过分号来结束变量声明，同时任何内容必须是有效的 Java 语句，例如：<%! int i=0; %>。注意，声明后面的分号不能省略。

### 3. 表达式

在 JSP 文档中，可以将表达式的结果直接输出到页面中。一般形式是：<% =表达式%>。

例如：

```
<% = i %>(输出变量 i 的值)
<% = "Hello" %>(输出字符串常量)
```

### 4. 代码段/脚本片段

JSP 代码段或脚本片段包含在“<% … %>”标记对内。当 Web 服务器响应请求时，这种 Java 代码就会运行。在脚本片段周围可能是纯粹的 HTML 或 XML 代码，在这些地方，代码片段可以创建条件执行代码，或只是调用另外一段代码。

例如，以下的代码组合使用表达式和脚本片段，分别按照 H1、H2、H3、H4 和 H5 标题样式，显示字符串“你好”，脚本片段并不局限于一行源代码中：

```
<% for (inti = 1; i<= 4; i++) { %>
<H<% = i %>>你好</H<% = i %>>
<% } %>
```

### 5. 注释

在文档中加入 HTML 注释，用户可以通过查看页面源代码来看到这些注释的内容。如果不想让用户看到注释内容，应将其嵌入到<%-- … --%>标记对中，一般形式是：

```
<%--注释内容--%>
```

**【例 6-13】**　编写一个 JSP 文档，显示网页的访问次数。

首先定义一个统计访问次数的文档，文档名为 mycount.jsp，内容如下：

```
<%! private int accessCount = 0; %>
<table border = "0" width = "100%" height = "60" bgcolor = "#FFFF00">
  <tr>
    <td width = "20%" height = "53">主机名：<% = request.getRemoteHost() %></td>
    <td width = "20%" height = "53">访问次数：<% = ++accessCount %></td>
    <td width = "60%" height = "53">当前时间：<% = new Date() %></td>
  </tr>
</table>
```

定义一个 JSP 页面，包含上述文件，输出一个随机数，mypage.jsp 文档内容如下：

```
<html>
<head>
<title>JSP 中的文件包含</title>
</head>
<body>
<%@ page import = "java.util.*" %>
<%! Random RdmNumber = new Random(); %>
<%@ include file = "mycount.jsp" %>
<%
    out.println(RdmNumber.nextInt(100)); //输出 100 以内的随机整数
%>
</body>
</html>
```

将上述文件保存在 d:/MyJSP 文件夹中,打开浏览器,输入 http://127.0.0.1/mypage.jsp 可以看到网页的输出结果。

### 6.6.3 JSP 内置对象

内置对象是由开发环境所定义的具有特定功能的对象,用户可以直接使用。在 JSP 脚本段中,可以访问这些隐含对象来与 JSP 网页中的可执行 Servlet 环境交互。JSP 中主要包含的内置对象有:

(1) request 对象,作用范围为整个页面。通过该对象可以获取客户端的输入信息。可以得到请求的参数、请求类型(GET、POST、HEAD 等)以及 HTTP headers(cookies、referer 等)等信息。这是一个 javax.servlet.HttpServletRequest 对象。

常用的方法有:getCookie()、getHeaders()、getAttribute()、getMethod()、getParameter()和 getParameterName()等。例如,要获取一个网页的 userName 参数的值,代码如下:

```
<% String name = request.getParameter("userName"); out.println(name); %>
```

通过上述代码可以获取用户访问网页时传入的参数值,包括页面中 Form 表单中各个属性域的值。例如,在网页 a.htm 中包含一个表单,其<form>的 action 属性指定表单处理页面为 b.jsp,在 b.jsp 中,可以通过 request.getParameter 方法来获取 a.htm 中的表单输入数据。

(2) response,作用范围为整个页面。它的作用是向客户端返回请求。返回请求信息时,输出流要进行缓存。它是一个 javax.servlet.HttpServletResponse 对象。

常用的方法有 addCookie()、addHeader()、sendError()和 sendRedirect()等。

(3) out,发送响应的输出流,作用是将结果输出到客户端。它最常用的方法有两个:print()和 println()。输出换行符使用 newLine()方法。

例如,不用表达式,可以直接访问隐含对象 out 来输出信息。

```
<% out.println("Hello"); %>
```

注意,不要同 Java 中的 System.out 对象混淆。

(4) session,作用范围为会话期间。Session 对象是 JSP 为每一个会话而建立的个人对象,可以存储及提供个别用户独享的永久或半永久信息。它是一个与 request 相关的 javax.servlet.http.HttpServletSession 对象。

所谓会话(session),是指一个用户和 Web 服务器之间的一次链接。当用户使用浏览器登录到 Web 服务器,并初次浏览一个 JSP 应用的某个网页开始,直到用户离开网站或超时未继续浏览该网站网页为止之间的浏览操作算作一次会话(Session)。

(5) application,该对象可存储并提供给一组 JSP 应用所有用户的共享信息,有效范围为构成该 JSP 应用的所有 JSP 页面。一般情况下,可以将 Application 对象作为一个存储许多共用对象的容器,这是一个 javax.servlet.ServletContext 对象。可通过 getServletConfig()、getContext()方法获得。

(6) config,用于传递在 Servlet 程序初始化时所需的信息。

(7) pageContext,该对象提供了对页面上的所有对象以及命名空间的访问,用于管理

网页属性。

(8) page,当前页面,相当于 Java 中的 this。

(9) exception,错误处理对象,用于处理捕捉到的异常。

**【例 6-14】** 编写一个 JSP 文档,完成一个登录界面,输入用户名和密码,如果输入 guest 则转移到一个默认的首页,如果不输入用户名,则重新回到该页,直到输入正确的用户名和密码,此时重定向到合法的网页。

文档(文档名 login. jsp)具体内容如下:

```
<!-- login.jsp -->
<html>
<head>
<title>login</title>
</head>
<body>
<%
String userName = request.getParameter("userName");
if (userName != null) {
  if (userName.equals("guest"))
      response.sendRedirect("http://www.sdu.edu.cn/");
  else
      response.sendRedirect("http://gsl.sdu.edu.cn/");
}
%>
<div align = "center">
 <center>
 <table border = "0" cellpadding = "0" width = "100 % " height = "100 % ">
 <tr>
 <td width = "100 % ">
      <table border = 2 cellspacing = 0 cellpadding = 0 width = 300 align = center>
      <form action = "login.jsp" method = "GET">
          <tr height = 50>
                  <td align = center style = "font - size:20;font - weight:bold;color:#
000075">用户登录</td>
          </tr>
          <tr><td style = "font - size:16">用户名<input type = text name = userName>
</td></tr>
          <tr><td style = "font - size:16">密 码<input type = text name = userPassword>
</td></tr>
          <tr><td align = center><input type = "submit" value = 登录></td></tr>
      </form>
      </table>
 </td>
 </tr>
 </table>
 </center>
</div>
</body>
</html>
```

将文件保存到 D:\MyJSP 文件夹中,在浏览器地址栏中输入:http://127.0.0.1/login.jsp,执行结果如图 6-8 所示。




图 6-8 一个 JSP 文档实现的登录界面

执行该程序时,如果是第一次执行,服务器首先对它进行转换、编译形成.class 字节码文件,接着运行该文件。第一次执行该文档时,用户名空,不执行开始处的 Java 代码段,此时输入用户名和密码,单击"登录"按钮,提交。由于表单的 action 属性为 login.jsp,即登录页面文件本身,此时再次执行该文档。此时,用户名不空,则将网页重定向到其他网页。如果用户名为空,则总是显示该网页。

**【例 6-15】** 编写一个 JSP 代码,显示在线人数。

首先编写一个统计会话人数的 Java 类,代码内容如下(文档名 SessionCounter.java):

```
package pub;
import javax.servlet.*;
import javax.servlet.http.*;
public class SessionCounter implements HttpSessionListener
{
  private static int activeSessions = 0;
  public void sessionCreated(HttpSessionEvent se)
  {
    activeSessions++;
  }
  public void sessionDestroyed(HttpSessionEvent se)
  {
    if (activeSessions > 0)
        activeSessions--;
  }
  public static int getActiveSessions()
  {
    return activeSessions;
  }
}
```

将上述 Java 类文件存储在 Web 应用的用户自定义类文件夹 classes 中的 pub 包(即一个子文件夹 pub)中,即存储到 d:\MyJSP\WEB-INF\classes\pub 中,然后编译该文件。

编写调用该 SessionCounter.java 的 JSP 文档，文档名为 myonline.jsp，内容如下：

```
<html>
<head>
</head>
<body>
<%@ page import = "pub.SessionCounter" %>
在线人数：<% = SessionCounter.getActiveSessions() %>
</body>
</html>
```

最后，修改 d:\MyJSP\WEB-INF\web.xml，在<web-app>…</web-app>内添加如下内容：

```
<!-- Listeners -->
<listener>
        <listener-class>
              pub.SessionCounter
        </listener-class>
</listener>
```

## 6.6.4　在 JSP 中使用 JavaBean

JavaBean 主要应用于 JSP 网页中，通过 JavaBean 更好地将业务逻辑代码和 JSP 的 HTML 标记进行分离，便于系统的维护。在 JSP 中使用 JavaBean 的一般形式是：

```
<jsp:useBean id = "实例名" class = "包.类" scope = "page|request|session| application" />
```

其中，id 指定一个 JavaBean 类的实例名，如果这个实例已经存在，将直接引用这个实例；如果这个实例不存在，将通过后面 class 参数中的定义新建一个类的实例。Class 参数设置存储 JavaBean 的路径和类的名称，例如 class＝"cards.NameCard"，则表明要使用 Web 应用根目录中"WEB-INF\classes\cards"下的一个 NameCard.class 文件，其中 cards 是包名，NameCard 为 JavaBean 对应的 Java 类名。Scope 用于定义 id 这个实例存在的范围，事实上这定义了这个实例所绑定的区域及其有效范围。

(1) page

这个 JavaBean 将存在于该 JSP 文件以及此文件中的所有静态包含文件中，直到页面执行完毕为止。等价于下面的 JSP 代码：

```
<%
  yourClass yourId;
  if (pageContext.getAttribute("yourId",PageContext.PAGE_SCOPE)==null)
  {
    yourId = new yourClass();
    pageContext.setAttribute("yourId",yourId,PageContext.PAGE_SCOPE);
  }
  else
    yourId = (yourClass)pageContext.getAttribute("yourId",PageContext.PAGE_SCOPE);
%>
```

(2) request

这个 JavaBean 将作为一个对象绑定于该页面的 request 中，即该 JavaBean 在该页面发出的请求中有效，等价于下面的 JSP 代码：

```
<%
  yourClass yourId;
  if (pageContext.getAttribute("yourId",PageContext.REQUEST_SCOPE)==null)
   {
    yourId = new yourClass();
    pageContext.setAttribute("yourId",yourId,PageContext.REQUEST_SCOPE);
  }
else
yourId = (yourClass)pageContext.getAttribute("yourId",PageContext.REQUEST_SCOPE);
%>
```

(3) session

这个 JavaBean 将作为一个对象绑定于 session 中，即该 JavaBean 在本地有效，等价于下面的 JSP 代码：

```
<%
  yourClass yourId;
  if (session.getAttribute("yourId")==null)
   {
    yourId = new yourClass();
    session.setAttribute("yourId",yourId);
  }
  else yourId = (yourClass)session.getAttribute("yourId");
%>
```

(4) application

这个 JavaBean 将作为一个对象绑定于 application 中，即该 JavaBean 在本应用中有效，等价于下面的 JSP 代码：

```
<%
  yourClass yourId;
  if (application.getAttribute("yourId")==null)
  {
    yourId = new yourClass();
    application.setAttribute("yourId",yourId);
  }
  else yourId = (yourClass)application.getAttribute("yourId");
%>
```

**【例 6-16】** 编写一个 JSP 页面，使用例 6-10 中定义的 JavaBean。

在 JSP 中调用 JavaBean 很简单，下面代码演示了例 6-10 中定义的 JavaBean 在一个 JSP 文档中的使用，文件名为 mycard.jsp，代码清单如下：

```
<html>
<body>
```

```
<%@ page language = "java" %>
<jsp:useBean id = "mycard" class = "cards.NameCard" scope = "application"/>
<%
  mycard.setName("Hao XW");
  mycard.setAddress("No.27 Shanda Southern Road,Jinan,China");
%>
姓名 :<% = mycard.getName() %><br>
地址 :<% = mycard.getAddress() %><br>
</body>
</html>
```

在上述的 JSP 页面中，调用了一个 JavaBean，即 cards.NameCard。将上述 JSP 文档保存到 Web 应用的主目录下，然后在浏览器地址栏中输入：http://127.0.0.1/mycard.jsp，则浏览器将显示运行结果。

### 6.6.5 JDBC 与数据库操作

在 Web 应用中，几乎都使用到了数据库来存储和管理系统数据。Java 中连接数据库的技术是 JDBC(Java Database Connectivity)，通过 java.sql 包中的类、接口和异常事件，可以对数据库进行操作。JDBC 的应用和传统的 ODBC(Open Database Connectivity，开放数据库互连)完全类似。

java.sql 包定义了访问数据库的接口和类，但是 JDBC API 不能直接访问数据库，必须依赖于数据库厂商提供的针对其具体的数据库产品的 JDBC 驱动程序。大多数的数据库管理系统都带有和 Java 相配的 JDBC 驱动程序，Java 程序通过 JDBC 驱动程序即可实现与数据库的相连，执行查询、提取数据等操作。使得 Java 程序能访问诸如 Oracle、Sybase、MS SQL Server、MS Access 和 MySQL 等数据库。要使用某种数据库管理系统，必须下载相应的 JDBC 数据库驱动程序，并复制到用户系统相应的文件夹中，即：用户系统根目录\WEB-INF\lib\。

#### 1. java.sql 包

JDBC API 主要位于 java.sql 包中，java.sql 包定义了所有的 JDBC 应用相关的类和方法，常用的类和接口是：

(1) java.sql.Driver 接口和 java.sql.DriverManager 类

DriverManager 类用来建立和数据库的连接以及管理 JDBC 驱动程序，常用方法包括：

- registerDriver(Driver driver)：在 DriverManager 中注册 JDBC 驱动程序。
- getConnection(String url,String user,String pwd)：建立和数据库的连接，返回 Connection 对象。
- setLoginTimeOut(int seconds)：设定等待数据库连接的最长时间。
- setLogWriter(PrintWriter out)：设定输入数据库日志的 PrintWriter 对象。

(2) java.sql.Connection 类

Connection 代表和数据库的连接，其常用方法如下：

- getMetaData()：返回数据库的 MetaData 数据。MetaData 数据包含了数据库的相关信息，例如当前数据库连接的用户名、使用的 JDBC 驱动程序、数据库允许的最大连接数、数据库的版本等等。
- createStatement()：创建并返回 Statement 对象。
- PrepareStatement(String sql)：创建并返回 PrepareStatement 对象。

在实际应用中，我们会遇到同时提交多个 SQL 语句，这些 SQL 语句要么全部成功，要么全部失败，如果其中一条提交失败，则必须撤销整个事务。为此，Connection 类还提供了 3 个控制事务的方法：

- setAutoCommit(boolen autoCommit)：设置是否自动提交事务，默认为自动提交。
- commit()：提交事务。
- rollback()：撤销事务。

(3) java. sql. Statement

Statement 用来执行静态 SQL 语句。例如，对于 insert、update、delete 语句，调用 executeUpdate(String sql)方法，而 select 语句可以调用 executeQuery(String sql)方法，executeQuery(String sql)方法返回 ResultSet 对象。

(4) java. sql. PrepareStatement

PrepareStatement 用于执行动态的 SQL 语句，即允许 SQL 语句中包含参数。使用方法为：

```
String sql  = "select col1 from tablename where col2 = ? And col3 = ?";
PrepareStatement perpStmt  = conn.preparestatement(sql);
perpStmt.setstring(1,col2Value);
perpStmt.setFloat(2,col3Value);
ResultSet rs  = perpStmt.executeQuery();
```

(5) java. sql. ResultSet

ResultSet 用来表示 select 语句查询得到的记录集，一个 Statement 对象在同一时刻只能打开一个 ResultSet 对象。通过 ResultSet 的 getXXX()方法来得到字段值。ResultSet 提供了 getString()、getFloat()、getInt()等方法。可以通过字段的序号或者字段的名字来指定获取某个字段的值。例如，在上例中 getString(0)、getString(col1)都可以获得字段 col1 的字符串值，通过 Interger 对象可以将字符串值转换为整数。

### 2. 在 JSP 页面中访问数据库

JSP 工作在三层结构的中间层，主要的功能是接受客户的请求，然后从数据库服务器获得信息，然后以 HTML 的形式返给客户浏览器。下面举例说明 JSP 访问 Microsoft SQL Server 2005 数据库的过程。

**【例 6-17】** 浏览网站留言数据库 gsl_messageboard。

设在 Web 应用系统中有一个留言板，供用户对站点进行留言，管理员可以查看用户留言并回复，用户可以查看留言及站点管理员对留言的回复。留言板数据库为 gsl_ messageboard。

(1) 在 Microsoft SQL Server 中，建立数据库 gsl_ messageboard，包含一个数据表 messageboard，结构定义如图 6-9 所示。

表 - dbo.messageboard* | 摘要

| | 列名 | 数据类型 | 允许空 |
|---|---|---|---|
| ▶ | title | nvarchar(50) | ☑ |
| | message | text | ☑ |
| | messagedate | datetime | ☑ |
| | author | nchar(10) | ☑ |
| | tel | varchar(50) | ☑ |
| | email | nvarchar(50) | ☑ |
| | feedback | text | ☑ |
| | feedbackdate | datetime | ☑ |
| | | | ☐ |

图 6-9　留言板数据库 messageboard 数据表结构定义

（2）在 Web 应用根目录下，创建相应的目录结构，即在 Web 应用根目录下创建 WEB-INF\文件夹，在 WEB-INF 下分别创建 classes 子文件夹和 lib 子文件夹，classes 文件夹存储用户定义的类，lib 文件夹存储应用系统所需要的驱动程序包.jar 文件。

在 WEB-INF\classes 文件夹下，可以进一步再定义一些子文件夹（即用户包），用于存储用户定义的 Java 类、JavaBean 源文件和.class 文件，例如，定义 WEB-INF\classes\pub 文件夹下，存储关于数据库操作的 JavaBean。

同时，还需要将 SQL Server 2005 jdbc 驱动 sqljdbc.jar 放置到用户系统的 lib 文件夹中，即 Web 应用根目录下的 WEB-INF\lib 文件夹下，该驱动程序可以从网上查找并下载。最后，在系统环境变量的 classpath 中添加驱动所在的目录，在 classpath 设置的后面，添加下面的路径：用户系统所在的驱动器：根目录\WEB-INF\lib\，例如：d:\mygpms\WEB-INF\lib\。

（3）在 WEB-INF\classes\pub 文件夹下，创建一个名为 db_gsl_messageboard.java 的文件，定义一个 JavaBean，封装有关数据库 gsl_messageboard 操作中的数据库连接、查询、数据更新以及断开数据库连接操作，内容如下：

```
package pub;
import java.sql.*;
public class db_gsl_messageboard
{
  String sConnStr = "jdbc:sqlserver://localhost:1433;DatabaseName = gsl_messageboard";
  String sDBDriver = "com.microsoft.sqlserver.jdbc.SQLServerDriver";
  String strUser = "sa";
  String strPassword = "sa";
  Connection conn = null;
  public ResultSet rs = null;
  private java.sql.Statement stmt = null;
  //定义类的默认构造函数
  public db_gsl_messageboard()
  {
    try { //加载数据库驱动程序(Microsoft SQL Server 2005)
        Class.forName(sDBDriver).newInstance();
        }
    catch (Exception e)
       {
        System.err.println("db_gsl_messageboard(): " + e.getMessage());
```

```
        }
    }
    //定义类的查询方法
    public ResultSet executeQuery(String sql)
    {
       rs = null;
       try {
           conn = DriverManager.getConnection(sConnStr, strUser, strPassword);
           Statement stmt = conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
                                                    ResultSet.CONCUR_READ_ONLY);
           rs = stmt.executeQuery(sql);
           }
        catch(SQLException ex)
         {
           System.err.println("aq.executeQuery: " + ex.getMessage());
          }
      return rs;
    }
    //定义类的更新数据记录方法
    public int executeUpdate(String sql)
    {
      int returnVal = -999;
      try {
          conn = DriverManager.getConnection(sConnStr, strUser, strPassword);
          Statement stmt = conn.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                                                   ResultSet.CONCUR_UPDATABLE);
          returnVal = stmt.executeUpdate(sql);
          }
      catch(SQLException ex)
          {
           System.err.println("aq.executeUpdate: " + ex.getMessage());
          }
         return returnVal;
    }
    //定义类的断开数据库方法
    public void disconnectToDB() throws java.sql.SQLException
    {
        if(rs! = null)
        {
           rs.close();
           rs = null;
         }
        if(stmt! = null)
        {
           stmt.close();
           stmt = null;
        }
        if(conn! = null)
        {
           conn.close();
           conn = null;
```

```
        }
    }
}
```

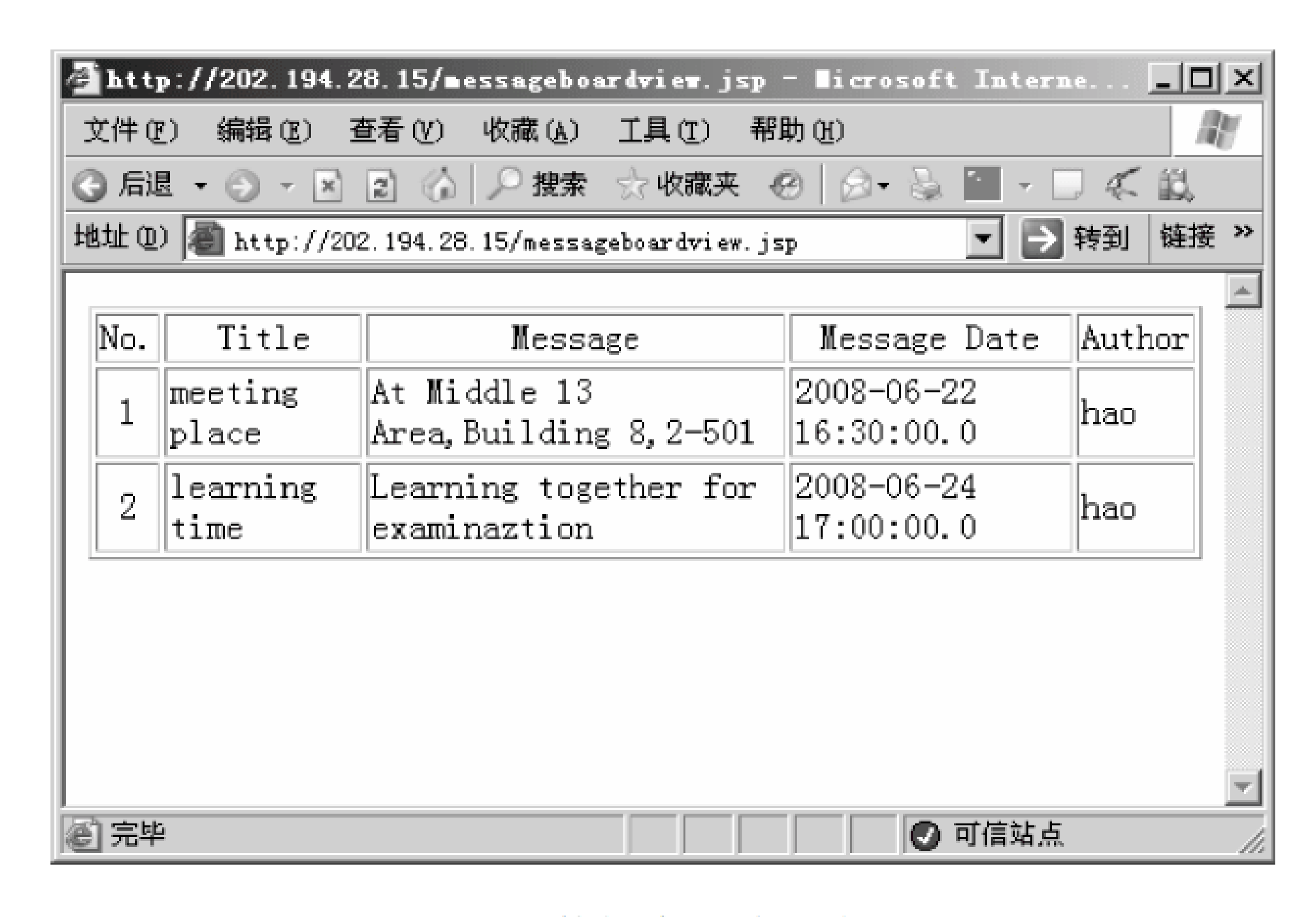
当上述文件编辑完成后，将 db_gsl_messageboard.java 文件保存，编译，则在 Web 应用根目录下的 WEB-INF\classes\pub 文件夹下生成 db_gsl_messageboard.class 文件，该 JavaBean 的.class 文件即可在 JSP 页面中被调用。

（4）使用 JavaBean。首先，编辑一个 JSP 页面文件 mesaageboardview.jsp，在 JSP 中调用以上编译好的 JavaBeans，其内容如下：

```
<%@ page contentType = "text/html;charset = GBK" %>
<%@ page language = "java" import = "java.sql.*" %>
<%@ page session = "true" %>
<jsp:useBean id = "myDBconn" scope = "page" class = "pub.db_gsl_messageboard" />
<%!
  ResultSet rs = null;
%>
<html>
<% request.setCharacterEncoding("GBK"); %>
<body>
<%
  // 获取数据库中数据表 messageboard 的记录
  ResultSet RS = myDBconn.executeQuery ("SELECT * FROM messageboard");
%>
<%! private int num = 1; %>
  <table border = "1">
  <tr align = "center">
  <td>No.</td> <td>Title</td><td>Message</td> <td>Message Date</td><td>
Author</td>
 </tr>
<%
   try
   {
      while (RS.next())
      { // 输出数据表中的所有记录
        out.print("<tr>");
        out.print("<td align = center>" + num + "</td>");
        out.print("<td>" + RS.getString("title") + "</td>");
        out.print("<td>" + RS.getString("message") + "</td>");
        out.print("<td>" + RS.getString("messagedate") + "</td>");
        out.print("<td>" + RS.getString("author") + "</td>");
        out.print("</tr>");
        num++;
      }
    }
  catch(Exception ex)
  {
      out.print(ex.getMessage());
  }
  finally
```

```
    {
        rs = null;
        myDBconn.disconnectToDB();
    }
%>
</table>
</body>
</html>
```

运行上述代码,显示如图 6-10 所示。



| No. | Title | Message | Message Date | Author |
|---|---|---|---|---|
| 1 | meeting place | At Middle 13 Area, Building 8, 2-501 | 2008-06-22 16:30:00.0 | hao |
| 2 | learning time | Learning together for examinaztion | 2008-06-24 17:00:00.0 | hao |

图 6-10 数据库浏览列表界面

在 Web 开发中,对数据库的访问通常被定义为一个 JavaBean,这样可以更好地实现业务逻辑和 JSP 中 HTML 代码的分离,同时也便于系统的维护,例如数据库升级后,原先的数据库访问语句可能不兼容。

## 6.6.6 JSP 与图形

经常看到网页上显示各种各样的图形,如:各种统计图、股市行情等。它们根据数据的变化来不断地更新图形,如何实现图形的绘制呢?

### 1. java.awt 包

java.awt 包是抽象窗口工具集(Abstract Window Toolkit,AWT),实现了可以跨平台的图形用户界面组件,可用于 Java 的 applet 和 applications 中,是 Java 图形用户界面编程的主要工具。它支持图形用户界面编程的功能,包括用户界面组件;图形和图像工具,包括形状、颜色和字体类;布局管理器,可以进行灵活的窗口布局而与特定窗口的尺寸和屏幕分辨率无关;数据传送类,可以通过本地平台的剪贴板来进行剪切和粘贴;事件处理模型等。

java.awt 包中提供了 GUI 设计所使用的类和接口,可分为组件(Component)、容器(Container)和布局管理器(LayoutManager)三个方面,主要的类和它们之间的关系如图 6-11 所示。

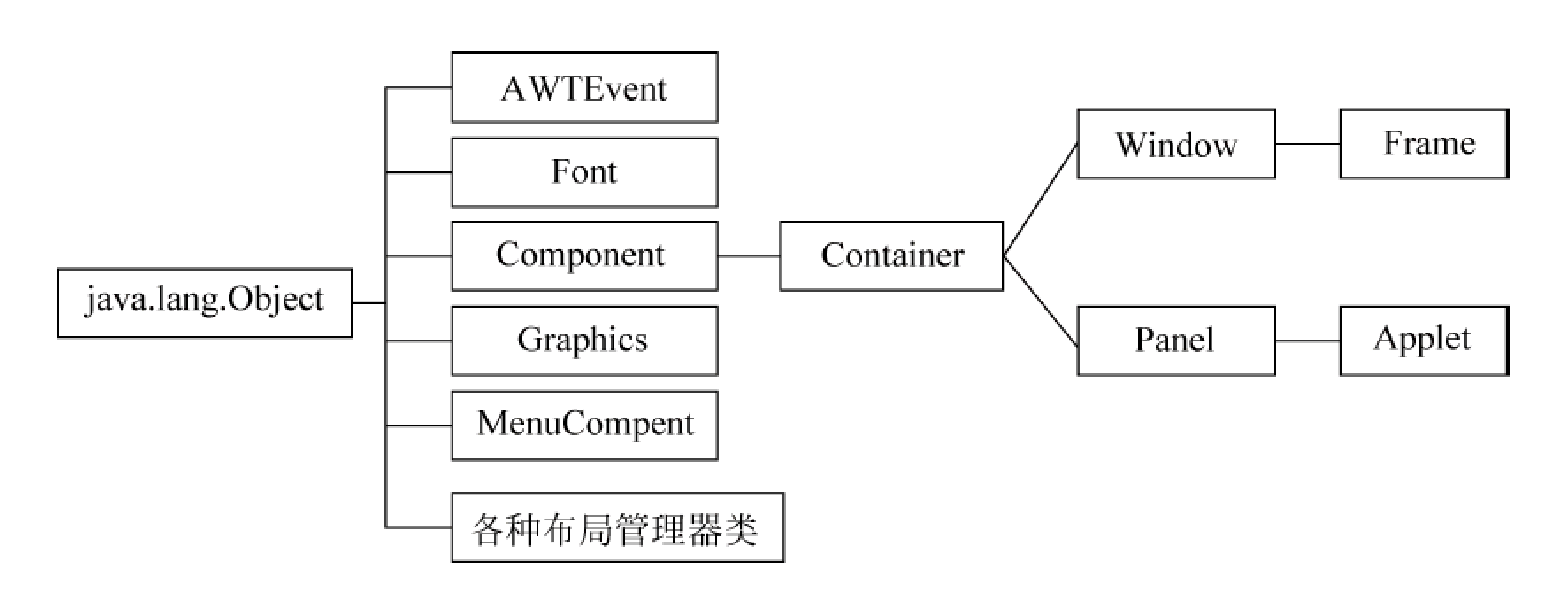



图 6-11　java. awt 包类结构图

(1) 组件

组件(Component)是 Java 的图形用户界面的最基本组成部分,是一个可以以图形化的方式显示在屏幕上并能与用户进行交互的对象,例如一个按钮、一个标签等。组件不能独立地显示出来,必须将组件放在一定的容器中才可以显示出来。

类 java. awt. Component 是许多组件类的父类,Component 类中封装了组件通用的方法和属性,如图形组件对象的大小、显示位置、前景色和背景色、边界、可见性等,因此许多组件类也就继承了 Component 类的成员方法和成员变量,相应的成员方法包括:

- getComponentAt(int x, int y)
- getFont()
- getForeground()
- getName()
- getSize()
- paint(Graphics g)
- repaint()
- update()
- setVisible(boolean b)
- setSize(Dimension d)
- setName(String name)等

(2) 容器

容器 java. awt. Container 是 Component 的子类,因此容器本身也是一个组件,具有组件的所有性质,但是它的主要功能是容纳其他组件和容器。一个容器可以容纳多个组件,并使它们成为一个整体。

使用容器可以简化图形化界面的设计,以整体结构来布置界面。所有的容器都可以通过 add()方法向容器中添加组件。有三种类型的容器:Window、Panel、ScrollPane,常用的类有 Frame、Panel、Applet。

- java. awt. Frame 类:Frame 类是 java. awt. Window 的子类,要生成一个窗口,通常是用 Window 的子类 Frame 来进行实例化,而不是直接用到 Window 类。Frame 的外观类似于在 Windows 操作系统下见到的窗口,有标题、边框、菜单、大小等等。每个 Frame 的对象实例化以后,都是没有大小和不可见的,因此必须调用 setSize( )来

设置大小,调用 setVisible(true)来设置该窗口为可见的。

- java.awt.Panel 类:一个 java.awt.Panel 对象隶属于一个窗口或框架对象,Panel 确定一个四边形,其他组件可以放入其中。Panel 必须放在 Window 之中(或 Window 的子类中)以便能显示出来。

**【例 6-18】** AWT 编程举例。

下面代码演示了利用 java.awt.Frame 和 java.awt.Panel 编写 GUI 界面的情况。

```
import java.awt.*;
public class FrameWithPanel extends Frame
{
  public FrameWithPanel(String str)
  {
   super(str);
  }
  public static void main(String args[])
  {
    FrameWithPanel fr = new FrameWithPanel("Hi");
    Panel pan = new Panel();
    fr.setSize(200,200);
    fr.setBackground(Color.red);          //框架 fr 的背景颜色设置为红色
    fr.setLayout(null);                   //取消布局管理器
    pan.setSize(100,100);
    pan.setBackground(Color.yellow);      //设置面板 pan 的背景颜色为黄色
    fr.add(pan);                          //用 add 方法把面板 pan 添加到框架 fr 中
    fr.setVisible(true);
  }
}
```

需要说明的是,AWT 在实际的运行过程中是调用所在平台的图形系统,因此同样一段 AWT 程序在不同的操作系统平台下运行所看到的图形系统是不一样的。例如,在 Windows 下运行,则显示的窗口是 Windows 风格的窗口;而在 UNIX 下运行时,则显示的是 UNIX 风格的窗口。

(3) 布局管理器(Layout Manager)

Java 为了实现跨平台的特性并且获得动态的布局效果,将容器内的所有组件安排给一个"布局管理器"负责管理,如将排列顺序,组件的大小、位置,当窗口移动或调整大小后组件如何变化等功能授权给对应的容器布局管理器来管理,而不使用直接设置组件位置和大小的方式。每个容器都有一个布局管理器,当容器需要对某个组件进行定位或判断其大小尺寸时,就会调用其对应的布局管理器。

不同的布局管理器使用不同算法和策略,容器可以通过选择不同的布局管理器来决定布局。布局管理器主要包括的类有:FlowLayout、BorderLayout、GridLayout、CardLayout、GridBagLayout。关于这些类的详细说明略。

在程序中安排组件的位置和大小时,应该注意以下两点:

第一,容器中的布局管理器负责各个组件的大小和位置,因此用户无法在这种情况下设置组件的这些属性。如果试图使用 Java 语言提供的 setLocation()、setSize()、setBounds()等方法,则都会被布局管理器覆盖。

第二，如果用户确实需要亲自设置组件大小或位置，则应取消该容器的布局管理器，方法为：setLayout(null)。

### 2. javax.swing 包

Java 扩展包 javax.swing 是新的图形界面类库，它继承了 java.awt 包，提供了多种图形界面组件，Swing 中包含了像标签页、表格、树、特殊边框、微调等各种新组件。Swing 中支持可插入观感(Pluggable look and feel ，PL&F)，支持用户定制桌面，更换新的颜色方案，让窗口系统适应特定的用户习惯和需要。Swing PL&F 体系结构使得同时定制 Swing 控件或控件组更加容易。

javax.swing 包中定义了大量的类和接口，常用的类和接口有：

- javax.swing，提供保证在任何平台上都有相同行为和表现的一套轻量级组件。
- javax.swing.border，提供在 Swing 组件周围绘制特殊边界的类集合和接口。
- javax.swing.colorchooser，包含能够被 JcolorChooser 组件所使用的类和接口集合。
- javax.swing.event，提供能够被 Swing 组件所触发的事件集。
- javax.swing.filechooser，包含能够被 JFileChooser 组件所使用的类和接口集合。
- javax.swing.plaf，提供一个接口和很多抽象类，用于为 Swing 提供可插入的 look-and-feel 能力。
- javax.swing.table，提供处理 javax.swing.Jtable 的类和接口集合。
- javax.swing.tree，提供处理 javax.swing.Jtree 的类和接口集合。
- javax.swing.text，提供处理可编辑的和不可编辑的文本组件的类和接口集合，支持文档的显示和编辑。
- javax.swing.text.html，提供 HTMLEditorKit 类和支持创建 HTML 文本编辑器的支持类集合，支持显示和编辑 HTML 文件。
- javax.swing.text.html.parser，提供默认的 HTML 解析器和相应的支持类。
- javax.swing.text.rtf，提供一个用于创建 Rich-Text-Format 文本编辑器的 RTFEditorKit 类，支持显示和编辑 RTF 文件。
- javax.swing.undo，为诸如文本编辑器一类的应用提供重复和撤销操作能力。

### 3. 使用 JavaBean 和 JSP 在网页上绘制图形

在网页中，如果要显示一幅变化的图形，可以将图形的绘制用 JavaBean 来完成，生成一幅 JPG 图片。然后，在 JSP 页面中调用上述 JavaBean，再通过 HTML 中的<img>标记来显示该图片。下面我们通过例子介绍在 JSP 上绘制和显示图形的方法。

**【例 6-19】** 利用 JSP+JavaBean 绘制和在网页上显示图形。

利用 JSP+JavaBean 在网页上绘制图形的一般步骤是：

(1) 在用户系统根目录下的 WEB-INF\classes 文件夹下，创建一个子文件夹，例如 pub，即定义一个包 pub，用于存储有关图形绘制的类和 JavaBeans。

(2) 在 pub 文件夹下，新建一个用来生成 jpg 文件的 Java Bean，文件名为 ChartGraph1.java，代码清单如下：

```
package pub;
```

```
//生成图片的 Java Bean
import java.io.*;
import java.util.*;
import com.sun.image.codec.jpeg.*;
import java.awt.image.*;
import java.awt.*;
public class ChartGraph1
{
     BufferedImage image;
     public void createImage(String fileLocation)
     {
          try {
               FileOutputStream fos = new FileOutputStream(fileLocation);
               BufferedOutputStream bos = new BufferedOutputStream(fos);
               JPEGImageEncoder encoder = JPEGCodec.createJPEGEncoder(bos);
               encoder.encode(image);
               bos.close();
          } catch(Exception e) {
               System.out.println(e);
          }
     }
    public void graphicsGeneration(int data[][],int groupnums,String filename)
    {
     int imageWidth = 720;            //图片的宽度
     int imageHeight = 230;           //图片的高度
     int columnWidth = 20;            //柱的宽度
     int columnHeight = 230;          //柱的最大高度
     int x = 4,i = 0;
     int bili = 1;                    //定义图形的绘制比例
     // 计算 data 中的最大数,这里假设为 100
     int maxnum = 100;
     if(imageHeight/maxnum==0)
          bili = 1;
     else
          bili = imageHeight/maxnum;  //实际值在图中的比例大小

     ChartGraphics chartGraphics = new ChartGraphics();
     chartGraphics.image = new BufferedImage(imageWidth, imageHeight,
                                        BufferedImage.TYPE_INT_RGB);
     Graphics graphics = chartGraphics.image.getGraphics();
     graphics.setColor(Color.white);
     graphics.fillRect(0,0,imageWidth,imageHeight);
     // graphics.drawRect(0,0,imageWidth - 1,imageHeight - 1);
     for(i = 0;i<groupnums;i++)
     {
          graphics.setColor(Color.red);
          graphics.fillRect(x + (i + 1) * columnWidth, columnHeight - data[i][0] * 2, columnWidth,
data[i][0] * 2);

          x = x + columnWidth;
          graphics.setColor(Color.pink);
          graphics.fillRect(x + (i + 1) * columnWidth, columnHeight - data[i][1] * 2, columnWidth,
data[i][1] * 2);
```

```
            x = x + columnWidth;
            graphics.setColor(Color.blue);
            graphics.fillRect(x + (i + 1) * columnWidth, columnHeight - data[i][2] * 2, columnWidth,
data[i][2] * 2);

            x = x + columnWidth;
        }
        chartGraphics.createImage("d:\\GSL3.0\\chart\\" + filename + ".jpg");
    }
}
```

说明：类 ChartGraph1 包含两个成员函数，一个用于创建文件，另一个用于图形绘制。绘制的图形文件扩展名为.jpg，并被存储在指定的文件夹中，供网页文件调用。

根据系统的功能，要绘制不同的图形，只需要修改 graphicsGeneration 函数及其参数即可，本例中绘制了五组柱状图形，每一组由三个并列的不同颜色的巨型块组成。

(3) 要绘制图形，需要传入相应的图形数据。图形数据通常来源于数据库(例如统计结果的图形绘制)，也可能来源于一些文本文件，或者是直接生成的数据。为了更好地将数据的获取和图形的显示分离，可以将数据获取定义为一个 JavaBean。

在 pub 文件夹下，编写一个读取数据的 JavaBean 类 GetData，根据不同的数据来源，读取数据的 JavaBean 不同，但都包含一个读取数据的函数，该函数的返回值通常是一个一维数组或多维数组，用于返回读取的数据。

类 GetData 定义的一般形式如下(文件名 GetData.java)：

```
package pub;
import java.io.*;
public class GetData
{
   int mydatas[] = new int[5];
   public int[] getDatas()   // 函数的返回类型为整数数组,返回读取的数据
   {
     try {
          RandomAccessFile randomAccessFile = new RandomAccessFile("d:\\GSL3.0\\mydata.
txt","r");
         for (int i = 0;i<5;i++)
         {
           mydatas[i] = Integer.parseInt(randomAccessFile.readLine());
         }
     }
     catch(Exception e)
     {
         System.out.println(e);
     }
     return mydatas;
   }
}
```

上述 JavaBean 从文本文件 C:\\GSL3.0\\mydata.txt 中读取图形绘制所用的数据。文件 mydata.txt 是一个文本文件，包含五行，每行一个整数。

(4) 在 Web 应用根目录下，编写一个 JSP 文档，来调用图形绘制的 Java 类，实现在网页

中的图形显示。设网页文件名为 chartview.jsp。mychart.jsp 代码清单如下：

```
<%@ page contentType="text/html;charset=GBK" %>
<%@ page import="java.util.*" %>
<jsp:useBean id="cg1" class="pub.ChartGraph1"/>
<jsp:useBean id="myData" class="GetData"/>
<%!
// int mydataary[] = new int[5];  // 定义一个存储图形数据的数组,保存获取数据 Java 类的
public int[]                      // getDatas()方法的返回值,该数组将作为实际参数传入图形
                                  // 绘制类相应的方法中
int[][] mydata  = new int[5][3];  // 本例中的图形绘制,需要使用一个二维数组存储图形数据
int groupnums = 5;                // 输出一组柱状图,每组为三个
Random RdmNumber = new Random();  // 生成随机数
%>
<%
// mydataary = myData.getDatas(); 通过 JavaBean 读取图形数据(一维数组)
for(int i = 0;i<5;i++)            // 为简化方便,本例中的图形数据是随机生成的,未使用 myData
     for(int j = 0;j<3;j++)
          mydata[i][j] = RdmNumber.nextInt(100);
// 调用 JavaBean 生成图形文件 mychart_1.jpg
cg1.graphicsGeneration(mydata,groupnums,"mychart_1");
%>
<html>
<body bgcolor="#AFC4ED" leftmargin="0" topmargin="0">
<img src="../chart/mychart_1.jpg">
</body>
</html>
```

在上述 chartview.jsp 文件中，为了简化代码，我们通过生成随机数对象为图形绘制生成数据，并没有使用我们定义的读取数据的 JavaBean 类 GetData。之所以讲解类 GetData，是为了使大家对图形绘制、图形数据获取有一个一般性的逻辑上的认识。

此外，根据绘制的图形不同，保存图形数据所使用的数据结构也不相同。在本例中，为了绘制一组柱状图，使用了一个二维数组 mydata。但是，在本例的 GetData 类的定义中，我们使用的是一个一维数组，这是由需要绘制的图形决定的。我们讲解这些不同的方法的目的是使大家对 Java 中的数组有个比较全面的了解，以便在以后的项目开发中应用。

(5) 运行

运行上述 chartview.jsp，显示如图 6-12 所示。

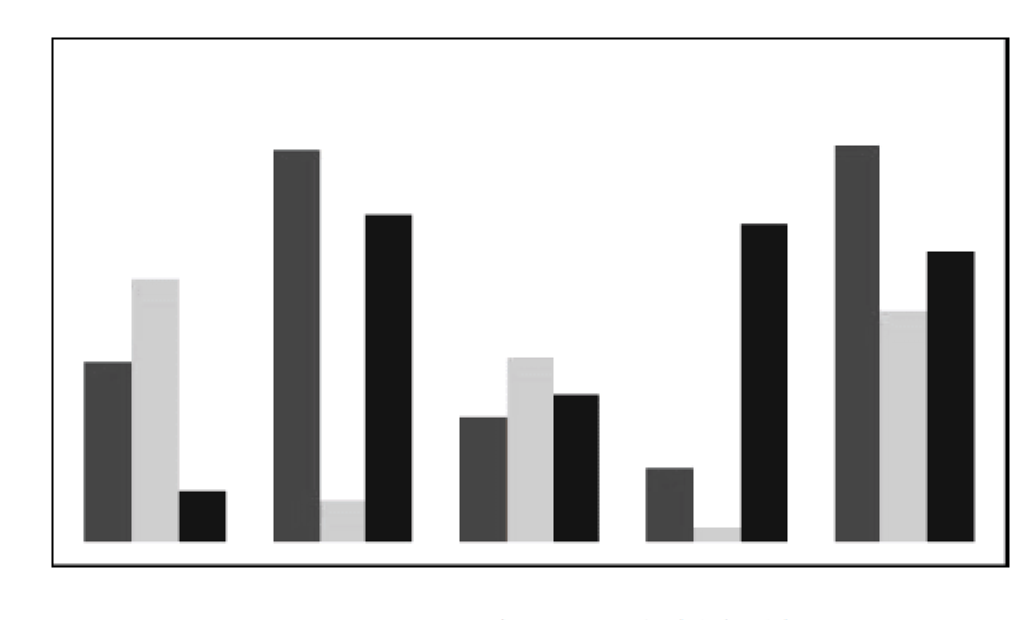


图 6-12　JSP 页面中图形绘制与输出页面

除了使用上述的方法外，还可以使用一些图表控件，如 Jfreechart 等，并基于这个控件开发自己的 JavaBean，采用开发的 JavaBean，可以将一些走势图生成 JPG 文件，然后在自己的网页里面调用它，从而可以用来显示各种走势图。

# 6.7　MVC 设计模式

实现软件系统的解耦是软件设计人员孜孜以求的目标，它对于保证系统的可维护性、可扩展性和开放性有着重要的意义。在 Web 开发中，主要的设计模式就是采用 MVC 设计模型，即：模型-视图-控制模型，从而将整个 Web 应用分成界面表现、业务逻辑和控制三个层次，从而保证系统更好的可维护性。

## 6.7.1　MVC 设计思想

模型-视图-控制（Model-View-Controller ，MVC）软件设计模式产生于 20 世纪 70 年代。近年来，随着 J2EE 的成熟，它正在成为在 J2EE 平台上推荐的一种设计模型，成为广大 Java 开发者非常感兴趣的设计模型。随着网络应用的快速增加，MVC 模式对于 Web 应用的开发无疑是一种非常先进的设计思想，无论选择哪种开发语言，无论应用多么复杂，MVC 都能为开发人员理解分析应用模型时提供最基本的分析方法，为软件系统设计提供清晰的设计框架，为软件开发提供规范的依据。

### 1. MVC 模型

MVC 设计思想将整个软件系统分成三个部分，即：模型（Model）、视图（View）和控制（Controller）。每个部分完成系统相应的功能，最大限度地将业务逻辑、界面表现进行分离，以保证系统的可维护性。

（1）视图（View）

视图代表用户交互界面，对于 Web 应用来说，可以概括为 HTML 界面，即网页。随着应用的复杂性和规模性，界面的处理也变得具有挑战性。一个应用可能有很多不同的视图，MVC 设计模式对于视图的处理仅限于视图上数据的采集和处理，以及用户的请求，而不包括在视图上的业务流程的处理。业务流程的处理交予模型（Model）处理。比如一个订单的视图只接受来自模型的数据并显示给用户，以及将用户界面的输入数据和请求传递给控制和模型。

（2）模型（Model）

模型就是业务流程/状态的处理以及业务规则的制定。业务流程的处理过程对其他层来说是暗箱操作，模型接受视图请求的数据，并返回最终的处理结果。业务模型的设计可以说是 MVC 最主要的核心。目前流行的 EJB 模型就是一个典型的应用例子，它从应用技术实现的角度对模型做了进一步的划分，以便充分利用现有的组件，但它不能作为应用设计模型的框架。

在 MVC 设计模式中，模型对应于用户的业务逻辑，因此，业务模型的设计是系统设计人员的重要任务。在业务模型的设计中，需要将应用的模型按一定的规则抽取出来。抽取

的层次很重要，抽象与具体不能隔得太远，也不能太近。MVC并没有提供模型的设计方法，而只要求组织管理这些模型，以便于模型的重构和提高重用性。

业务模型还有一个很重要的模型就是数据模型。数据模型主要指实体对象的数据保存（持续化），比如将一张订单保存到数据库，从数据库获取订单。我们可以将这个模型单独列出，所有有关数据库的操作只限制在该模型中。

(3) 控制(Controller)

控制可以理解为从用户接收请求，将模型与视图匹配在一起，共同完成用户的请求。控制层的作用很明显，它就是一个分发器，选择什么样的模型，选择什么样的视图，可以完成什么样的用户请求。控制层并不做任何的数据处理。例如，用户点击一个连接，控制层接受请求后，并不处理业务信息，它只把用户的信息传递给模型，告诉模型做什么，选择符合要求的视图返回给用户。因此，一个模型可能对应多个视图，一个视图可能对应多个模型。

**2. MVC模型的优点**

在Web应用的开发中，初始的开发模板往往是一种混合层的数据编程。例如，在ASP、JSP页面中直接向数据库发送请求并用HTML显示，业务逻辑和显示混在一起。这样的开发方式开发速度往往比较快，但由于数据页面的分离不是很直接，因而很难体现出明显的业务模型，难以实现业务模型的重用，难以应对用户的变化性需求。

采用MVC设计模式进行软件系统的开发，充分实现业务逻辑和显示的分离，主要优点如下：

(1) 一个模型可以对应多个显示视图。在目前用户需求的快速变化下，可能有多种方式访问应用的要求。例如，订单模型可能有本系统的订单，也有网上订单，或者其他系统的订单，但对于订单的处理都是一样的，也就是说订单的处理是一致的。按MVC设计模式，一个订单模型以及多个视图即可解决问题。这样减少了代码的复制，即减少了代码的维护量，一旦模型发生改变，也易于维护。

(2) 一个应用的业务流程或者业务规则的改变只需改动MVC的模型层。此外，由于模型返回的数据不带任何显示格式，因而这些模型也可直接应用于接口的使用。由于控制层把不同的模型和不同的视图组合在一起完成不同的请求，因此，控制层可以说是可以实现用户请求权限的概念。

采用MVC设计模式，整个应用被分成三层，每个文件的功能明确，这必然带来代码文件数量上的增加。但是，各个文件的功能清晰，只要合理地进行文件的分类管理，将使Web应用开发更有利于工程化。

### 6.7.2 使用JavaBeans实现业务逻辑

在MVC设计模式下，业务逻辑应该从JSP页面中进行有效的分离，在技术上，可以通过JavaBeans来实现。也就是说，在业务逻辑设计上，把一个个的业务逻辑编写成对应的JavaBean，然后将这些JavaBeans存储在Web应用主目录下WEB-INF\classes\中不同业务模型文件夹中，这些文件夹分别对应一个个Java类包(Package)。

下面通过Web应用中用户注册功能的设计来说明MVC中的模型实现。用户注册是

Web 系统最常见的功能，主要是获取用户信息，然后为用户创建一个系统账户和分配用户权限。涉及的主要技术问题是客户端的数据有效性验证和服务端的数据库操作，因此，该业务包括了两个文件，一个用于用户交互的 html 文件，一个用于数据库操作的 JSP 文件。

（1）视图设计视图即用户的交互界面，用户界面的设计包括两个方面，一方面是功能设计，另一方面是平面设计，两者需要有机的结合。

为描述简单起见，我们给出一个简化的用户注册界面，如图 6-13 所示。

GSL3.0 用户注册 jcjy.sdu.edu.cn
系统首页　退出系统
欢迎注册GSL3.0用户，请您填写下面的资料，有 * 标记的必须填写
输入用户名
* 用户名：　用户名由英文字母a~z，数字0~9，特殊字符组成。
设置密码
* 密码：　密码长度6~20位，由字母，数字，特殊字符组成。
* 请再次输入密码：
密码保护设置　为了保障您的帐号安全，请至少选择两种方式进行填写并牢记。
1、常用邮箱：　遇到问题或忘记密码，可通过此邮箱联系取回。
2、手机号码：　通过手机可以取回密码。
个人信息
* 姓名：　请提供真实姓名，以方便我们今后联系。
* 性别：　男　女
* 职业：　-------请您选择职业-------
* 单位：
* 联系方式：　请提供电话或手机，以方便我们今后联系。
* 邮箱：　请提供常用邮箱。
通信地址：
邮编：
服务条款
尊敬的用户，欢迎您注册并使用GSL3.0系统服务。
在注册及使用前请您仔细阅读如下服务条款：
1、该系统的来宾主要是指校外教师、学生以及本校非教学计划中的学生。
2、注册来宾帐户之后，可以拥有一定的权限访问本站点。
我同意以上服务条款，并提交申请
Copyright © http://jcjy.sdu.edu.cn All Rights Reserved.
山东大学计算机科学与技术学院 Email:hxw@sdu.edu.cn

图 6-13　用户注册界面

用户界面平面设计完成用户的人机交互设计，一般是一个 HTML 页面，通常会包含一些 JavaScript 脚本程序，相关的数据操作则通过页面中表单的 action 参数设定。

用户注册界面对应的 HTML 文件名为 new_user.html，代码清单如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
```

```
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<link rel="stylesheet" type="text/css" href="styles/newuser.css" />
<title>GSL3.0 新用户注册</title>
<script language="javascript">
 function on_submit()
 {//验证数据的合法性
  if (myform.username.value =="")
   {
     alert("用户名不能为空,请输入用户名!");
     myform.username.focus();
     return false;
   }
  if (myform.password.value =="")
   {
     alert("用户密码不能为空,请输入用户密码!");
     myform.password.focus();
     return false;
    }
 if (myform.password.value! = myform.repassword.value)
   {
     alert("两次密码输入不一致,请重新输入!");
     myform.password.focus();
     return false;
    }
  if (myform.email1.value.length==0)
   {
       alert("请输入 Email!");
       myform.email1.focus();
       return false;
    }
   else
    {
       for (i=0; i<myform.email1.value.length; i++)
              if (myform.email1.value.charAt(i)=="@")
                  break;
       if (i==myform.email1.value.length)
       {
              alert("非法 Email 地址, 请重新输入!");
              myform.email1.focus();
              return false;
       }
     }
  }
</script>
</head>
<body>
<div id="main">
<div id="banner">
```

```
<table id="banner-table" cellpadding="0" cellspacing="0" border="0">
<tr>
  <td width="200px" background="../images/newuser/logo.jpg"></td>
  <td width="20px"></td>
  <td width="6px" background="images/newuser/yuanjiao_left.jpg"></td>
  <td width="510px" bgcolor="#0163A2">
  <div id="banner-table_note">
  <a href="v3_indexguest.jsp" target="_self">系统首页</a>  <a href="v3_
loginout.jsp" target="_self">退出系统</a>
  </div>
  </td>
  <td width="4px" background="images/newuser/yuanjiao_right.jpg"></td>
</tr>
</table>
</div>
<div id="info">
<form name="myform"action="new_usersave.jsp" method="post" onSubmit="return on_submit()">
<input type="hidden" value="" name="id">
<table id="info_table" cellpadding="0" cellspacing="0">
<tr>
  <td id="row1">
  <div id="jiantou"></div>
  <div id="row1_note">欢迎注册 GSL3.0 用户,请您填写下面的资料,有 <font class="rad"> *
</font> 标记的必须填写</div>
  </td><!--上面颜色条-->
</tr>
<tr>
  <td id="row2">
  <table class="password" cellpadding="0" cellspacing="0" border="0">
  <tr><td class="mimashezhi" colspan="3"><strong>输入用户名</strong></td></tr>
  <tr><td class="kong" colspan="3"></td></tr>
  <tr>
      <td width="20%" class="mima_note"><font class="rad"> * </font> 用户名:
</td>
      <td width="40%" class="mima_note1"><input class="pas" type="text" name=
"username"/></td>
      <td class="mima_note2">用户名由英文字母 a~z、数字 0~9、特殊字符组成。</td>
  </tr>
  <tr><td class="kong" colspan="3"></td></tr>
  <tr><td class="line" colspan="3"></td></tr>
  </table>
  </td>
</tr>
<tr>
  <td id="row2">
  <table class="password" cellpadding="0" cellspacing="0" border="0">
  <tr><td class="mimashezhi" colspan="3"><strong>设置密码</strong></td></tr>
  <tr><td class="kong" colspan="3"></td></tr>
  <tr>
      <td width="20%" class="mima_note"><font class="rad"> * </font> 密码:</td>
      <td width="40%" class="mima_note1"><input class="pas" type="password" name=
```

```
"password"/></td>
        <td class = "mima_note2">密码长度 6～20 位,由字母、数字、特殊字符组成。</td>
  </tr>
  <tr>
        <td width = "20 % " class = "mima_note"><font class = "rad"> * </font> 请再次输入
密码：</td>
        <td class = "mima_note1"><input class = "pas" type = "password" name = "repassword"/>
</td>
        <td class = "mima_note"></td>
  </tr>
  <tr><td class = "kong" colspan = "3"></td></tr>
  <tr><td class = "line" colspan = "3"></td></tr>
  </table>
  </td>
</tr>
<tr>
  <td id = "row2">
  <table class = "password" cellpadding = "0" cellspacing = "0" border = "0">
  <tr>
        <td class = "mimashezhi"><strong>密码保护设置</strong></td>
        <td class = "mimashezhi" colspan = "2"><font class = "rad">为了保障您的账号安全,
请至少选择两种方式进行填写并牢记。</font></td>
  </tr>
  <tr><td class = "kong" colspan = "3"></td></tr>
  <tr>
        <td width = "20 % " class = "mima_note"><font class = "rad">1、</font>常用邮箱：
</td>
        <td width = "40 % " class = "mima_note1"><input class = "pas" type = "text" name =
"email"/></td>
        <td class = "mima_note2">遇到问题或忘记密码,可通过此邮箱联系取回。</td>
  </tr>
  <tr>
        <td width = "20 % " class = "mima_note"><font class = "rad">2、</font>手机号码：
</td>
        <td width = "40 % " class = "mima_note1"><input class = "pas" type = "text" name =
"cellphone"/></td>
        <td class = "mima_note2">通过手机可以取回密码。</td>
  </tr>
  <tr><td class = "kong" colspan = "3"></td></tr>
  <tr><td class = "line" colspan = "3"></td></tr>
  </table>
  </td>
</tr>
<tr>
  <td id = "row2">
  <table class = "password" cellpadding = "0" cellspacing = "0" border = "0">
  <tr><td class = "mimashezhi" colspan = "3"><strong>个人信息</strong></td></tr>
  <tr><td class = "kong" colspan = "3"></td></tr>
  <tr>
        <td width = "20 % " class = "mima_note"><font class = "rad"> * </font> 姓名：</td>
        <td width = "40 % " class = "mima_note1"><input class = "pas" type = "text" name =
```

```
"name"/></td>
        <td class = "mima_note2">请提供真实姓名,以方便我们今后联系。</td>
  </tr>
  <tr>
        <td width = "20 % " class = "mima_note"><font class = "rad"> * </font> 性别:</td>
        <td width = "40 % " class = "mima_note1">
        <input type = "radio" name = "sex" value = "男" class = "radio">男
        <input type = "radio" name = "sex" value = "女" class = "radio">女
        </td>
        <td class = "mima_note2"></td>
  </tr>
  <tr>
        <td width = "20 % " class = "mima_note"><font class = "rad"> * </font> 职业:</td>
        <td width = "40 % " class = "mima_note1">
        <select name = "userclass" class = "select">
        <option value = "0">------请您选择职业------</option>
        <option value = "1">教师</option>
        <option value = "2">学生</option>
        <option value = "3">其他</option>
        </select>
        </td>
        <td class = "mima_note2"></td>
  </tr>
  <tr>
        <td width = "20 % " class = "mima_note"><font class = "rad"> * </font> 单位:</td>
        <td width = "40 % " class = "mima_note1"><input class = "pas" type = "text" name =
"department"/></td>
        <td class = "mima_note2"></td>
  </tr>
  <tr>
        <td width = "20 % " class = "mima_note"><font class = "rad"> * </font> 联系方式:
</td>
        <td width = "40 % " class = "mima_note1"><input class = "pas" type = "text" name =
"phone"/></td>
        <td class = "mima_note2">请提供电话或手机,以方便我们今后联系。</td>
  </tr>
  <tr>
        <td width = "20 % " class = "mima_note"><font class = "rad"> * </font> 邮箱:</td>
        <td width = "40 % " class = "mima_note1"><input class = "pas" type = "text" name =
"email1"/></td>
        <td class = "mima_note2">请提供常用邮箱。</td>
  </tr>
  <tr>
        <td width = "20 % " class = "mima_note">通信地址:</td>
        <td width = "40 % " class = "mima_note1"><input class = "pas" type = "text" name =
"address"/></td>
        <td class = "mima_note2"></td>
  </tr>
  <tr>
        <td width = "20 % " class = "mima_note">邮编:</td>
        <td width = "40 % " class = "mima_note1"><input class = "pas" type = "text" name =
```

```
"post"/></td>
        <td class = "mima_note2"></td>
  </tr>
  <tr><td class = "kong" colspan = "3"></td></tr>
  <tr><td class = "line" colspan = "3"></td></tr>
  </table>
  </td>
</tr>
<tr>
  <td id = "row2">
  <table class = "password" cellpadding = "0" cellspacing = "0" border = "0">
  <tr><td class = "mimashezhi" colspan = "3"><strong>服务条款</strong></td></tr>
  <tr><td class = "kong" colspan = "3"></td></tr>
  <tr>
       <td colspan = "3" class = "fuwu">
       <textarea name = "title" class = "fuwu">
      尊敬的用户,欢迎您注册并使用 GSL3.0 系统服务。
      在注册及使用前请您仔细阅读如下服务条款:
      1. 该系统的来宾主要是指校外教师、学生以及本校非教学计划中的学生。
      2. 注册来宾账户之后,可以拥有一定的权限访问本站点。
      3. 因系统管理的需要,管理员有权删除来宾账户。
       </textarea>
       </td>
  </tr>
  <tr><td class = "kong" colspan = "3"></td></tr>
<tr>
  <td class = "mimashezhi1" colspan = "3"><input type = "submit" value = "我同意以上服务条
款,并提交申请" class = "menu"></td>
  </tr>
  <tr><td class = "kong" colspan = "3"></td></tr>
  </table>
  </td>
</tr>
</table>
</form>
</div>
<div id = "foot">
<table id = "foot - table" cellpadding = "0" cellspacing = "0" border = "0">
<tr>
  <td width = "5px"><img src = "../images/newuser/yuanjiao_left1.jpg" /></td>
  <td bgcolor = "#0163A2" class = "foot_center">
  <table id = "foot_table1" cellspacing = "0" cellpadding = "0" border = "0">
<tr>
        <td class = "foot_text">Copyright &copy; http://jcjy.sdu.edu.cn All Rights
Reserved.</td>
  </tr>
  <tr> <td class = "foot_text">山东大学计算机科学与技术学院 Email:hxw@sdu.edu.cn</td>
  </tr>
  </table>
  </td>
  <td width = "4px"><img src = "../images/newuser/yuanjiao_right1.jpg" /></td>
```

```
</tr>
</table>
</div>
</div>
</body>
</html>
```

在上述 HTML 页面中，定义了页面对应的样式文件 newuser.css，其中定义了表格中应用到的众多样式，这些样式在<table>等标记中通过 ID 属性来引用，增加了页面维护的灵活性。

（2）业务模型

在视图设计完成后，接下来是为页面添加程序代码，以实现页面的功能。页面功能反映了系统的业务模型，根据 MVC 设计模式，业务模型应通过 JavaBean 实现，这些 JavaBean 通过 JSP 页面来调用。

在 new_user.html 中，当用户单击"同意服务条款"按钮后，执行 new_usersave.jsp 文件，完成数据库的查询和增加记录操作，代码清单如下：

```
<%@ page contentType="text/html;charset=gb2312" %>
<%@ page session="true" %>
<%@ page import="java.sql.*" %>
<%@ page import="java.util.*,javax.mail.*" %>
<%@ page import="javax.mail.internet.*" %>
<jsp:useBean id="workE" scope="page" class="pub.connaccounts" />
<jsp:useBean id="workd" scope="page" class="pub.mytime"/>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-transitional.dtd">
<%!
ResultSet rs = null;
String content="";
String time="";
int flag=0;
%>
<%!
public String codeToString(String str)
{//处理中文字符串的函数
  String s=str;
  try
    {
    byte tempB[]=s.getBytes("ISO-8859-1");
    s=new String(tempB);
    return s;
   }
  catch(Exception e)
   {
    return s;
   }
}
%>
<% //接收客户端页面 new_user.html 提交的表单数据
```

```
String username = codeToString(request.getParameter("username"));
String password = codeToString(request.getParameter("password"));
String email = codeToString(request.getParameter("email"));
String email1 = codeToString(request.getParameter("email1"));
String department = codeToString(request.getParameter("department"));
String cellphone = codeToString(request.getParameter("cellphone"));
String address = codeToString(request.getParameter("address"));
String phone = codeToString(request.getParameter("phone"));
String userclass = codeToString(request.getParameter("userclass"));
String sex = codeToString(request.getParameter("sex"));
String name = codeToString(request.getParameter("name"));
String post = codeToString(request.getParameter("post"));
time = workd.getTime();
content = "尊敬的用户:" + "\n" + "\n" + "您好!" + "\n" + "\n" + "欢迎您注册 GSL3.0 系统,您的信息
已经提交,请等待管理员的审核。" + "\n" + "\n" + "以下是您的注册信息,请注意保管:" + "\n" + "用
户名:" + username + "\n" + "密码:" + password + "\n" + "\n" + "找回密码邮箱:" + email + "\n" + "手
机:" + cellphone + "\n" + "姓名:" + name + "\n" + "性别:" + sex + "\n" + "单位:" + department + "\
n" + "联系方式:" + phone + "\n" + "邮箱:" + email1 + "\n" + "\n" + "请保管好个人信息,祝您使用
愉快!" + "\n" + "\n" + time;
%>
<%
// 构造查询记录 SQL 语句
String sqlString1 = "";
sqlString1 = "SELECT * FROM guest_info where username = '" + username.trim() + "'";
// 构造修改记录 SQL 语句
String sqlString2 = "";
sqlString2 = " insert into guest _ info ( username, password, email, cellphone, name, sex,
userclass,department,phone,email1,address,post,if_pass,add_time) values('" + username + "',
'" + password + "','" + email + "','" + cellphone + "','" + name + "','" + sex + "','" + userclass + "',
'" + department + "','" + phone + "','" + email1 + "','" + address + "','" + post + "','0','" + time + "')";
try
{
    rs = workE.executeQuery(sqlString1);
    if(rs.next())
    {
        flag = 0;
    }
    else
    {
        workE.executeUpdate(sqlString2);
        flag = 1;
    }
}
catch(Exception ex)
{
    out.print(ex.getMessage());
}
finally
{
    rs = null;
    workE.disconnectToDB();
```

```
}
%>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<link rel="stylesheet" type="text/css" href="../scripts/newuser1.css" />
<title>GSL3.0 新用户注册反馈</title>
</head>
<body>
<div id="main">
<div id="banner">
<table id="banner-table" cellpadding="0" cellspacing="0" border="0">
<tr>
 <td width="200px" background="images/newuser/logo.jpg"></td>
 <td width="20px"></td>
 <td width="6px" background="images/newuser/yuanjiao_left.jpg"></td>
 <td width="510px" bgcolor="#0163A2">
 <div id="banner-table_note">
 <a href="v3_indexguest.jsp" target="_self">系统首页</a>  <a href="v3_
loginout.jsp" target="_self">退出系统</a>
 </div>
 </td>
 <td width="4px" background="images/newuser/yuanjiao_right.jpg"></td>
</tr>
</table>
</div>
<div id="info">
<table id="info_table" cellpadding="0" cellspacing="0">
<tr>
 <td id="row1">
 <div id="jiantou"></div>
 <div id="row1_note">欢迎注册 GSL3.0 用户</div>
 </td><!--上面颜色条-->
</tr>
<tr>
 <td id="row2">
 <table class="password" cellpadding="0" cellspacing="0" border="0">
 <tr><td class="mimashezhi" colspan="3"> </td> </tr>
 <tr><td class="kong" colspan="3"></td> </tr>
<%
if (flag==0)
{
%>
<tr><td width="100%" class="mima_note3"><b>该用户名已被申请！</b></td></tr>
<tr><td width="100%" class="mima_note3"></td></tr>
<tr>
    <td width="100%" class="mima_note4"><a href="javascript:history.back(-1)">请
重新填写</a></td>
</tr>
<%
}
```

```
else
{
%>
 <tr><td width="100%" class="mima_note3"><b>注册成功!</b></td></tr>
 <tr><td width="100%" class="mima_note3"></td></tr>
 <tr>
      <td width="100%" class="mima_note5">1.您的信息已被提交,请等待管理员的审核。
</td>
</tr>
<tr>
<td width="100%" class="mima_note5">2.系统已将注册信息发送到您的邮箱中,请注意查收。
</td>
</tr>
<tr>
<td width="100%" class="mima_note5"><a href="v3_indexguest.jsp">3.点击返回登录界
面</a></td>
</tr>
<%
}
%>
 <tr><td class="kong" colspan="3"></td></tr>
 <tr><td class="line" colspan="3"></td></tr>
 </table>
 </td>
</tr>
</table>
</div>
<div id="foot">
<table id="foot-table" cellpadding="0" cellspacing="0" border="0">
<tr>
 <td width="5px"><img src="../images/newuser/yuanjiao_left1.jpg" /></td>
 <td bgcolor="#0163A2" class="foot_center">
 <table id="foot_table1" cellspacing="0" cellpadding="0" border="0">
 <tr>
      <td class="foot_text">Copyright &copy; http://jcjy.sdu.edu.cn All Rights Reserved.
</td>
 </tr>
 <tr>
      <td class="foot_text">山东大学计算机科学与技术学院 Email:hxw@sdu.edu.cn</td>
 </tr>
 </table>
 </td>
 <td width="4px"><img src="../images/newuser/yuanjiao_right1.jpg" /></td>
</tr>
</table>
</div>
</div>
</body>
</html>
```

在本例中,用户注册界面主要功能是完成数据库的操作,包括连接数据库、数据库查询、

添加记录等，使用了两个 JavaBean 类，分别为 pub.connaccounts 和 pub.mytime，前者用于连接用户信息数据库，后者用于处理 Java 中的时间。pub.connaccounts 和 pub.mytime 代码和 6.6.5 节的介绍非常类似，在此不再重复。

用户注册成功后，显示信息反馈界面，如图 6-14 所示。




图 6-14　用户注册成功后的信息反馈界面

### 6.7.3　使用 CSS 控制显示视图

传统的页面设计主要是 Table 来进行布局。现在，在页面设计中，为了使页面具有更好的灵活性和可维护性，通常采用图层(div)、Table 和 CSS 技术来设计。因此，对于上述的用户注册界面，可以设计对应的 CSS 文件，来定义所用的 HTML 元素的显示外观。

对应用户注册页面 new_user.htm 和用户注册反馈页面 new_usersave.jsp，都定义了相应的 CSS 文件，即 newuser.css 和 newusersave.css，详细内容略。

## 6.8　综合举例——在线聊天程序

在本书的最后，我们将通过在线聊天程序的开发来对所学的知识进行一个综合的应用。在线聊天是许多网站都提供的一种功能，它包含了服务端开发、客户端开发、Ajax 技术、JavaBeans 以及数据库的应用，涉及的内容比较全面，相对容易理解。

### 6.8.1　系统分析

在线聊天程序系统中，用户可分为管理员用户和普通用户，普通用户可以选择聊天对象，发送聊天信息，管理员可以对在线用户发出警告，甚至将用户踢出聊天室。

聊天程序设计包括两个方面，一方面是客户端的用户界面，另一方面是服务端的数据库操作，用于记录用户的聊天记录。此外，为避免用户聊天信息更新带来的页面闪烁，我们采用 Ajax 技术，来进行客户端和服务端的异步数据传输。

## 6.8.2 客户端设计

聊天程序主要涉及一个用户界面，这就是客户端用户界面。客户端用户界面通常分成几个区域：用户列表、聊天信息输入区域、聊天记录显示区域。上述用户界面可以通过HTML的表格和图层来布局，常见的用户界面如图6-15所示。




图6-15 聊天系统客户端用户界面

聊天系统客户端界面分成三个区域，左侧为用户信息显示区域，显示用户个人信息和在线用户列表。右侧上部为聊天记录的显示区域，下部为用户输入区。下面介绍各个部分的实现代码，包括聊天主界面代码清单。

### 1. 客户端用户界面 chat_frame.jsp

聊天程序的主界面页面为chat_frame.jsp，采用在Table中插入IFRAME的方式，实现不同区域的功能，chat_frame.jsp代码清单如下：

```
<%@ page contentType = "text/html;charset = GBK" %>
<!DOCTYPE html PUBLIC " - //W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1 - transitional.dtd">
<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
<meta http - equiv = "Content - Type" content = "text/html; charset = gb2312" />
<link rel = "stylesheet" type = "text/css" href = "scripts/chat.css">
</head>
```

```
<body>
<div id="main">
<table id="banner" cellpadding="0" cellspacing="0" border="0">
  <tr>
      <td width="3px" background="images/open/banner_left.jpg"></td>
      <td width="871px"> </td>
      <td width="4px" background="images/open/banner_right.jpg"></td>
    </tr>
</table>
<table id="main_table" cellpadding="0" cellspacing="0">
    <tr>
      <td rowspan="2" id="left">
            <iframe name=left src="chat_left.jsp" frameBorder=0 style="z-index: 2;
visibility: inherit; width: 100%; height: 100%"></iframe>
    </td>
    <td id="right">
      <iframe name=right src="chat_client.html" frameBorder=0 scrolling=no style="z-
index: 2; visibility: inherit; width: 100%; height:100%"></iframe>
      </td>
    </tr>
    <tr>
      <td id="bottom">
          <iframe name=bottom src="chat_input.jsp" frameBorder=0 scrolling=no style="z-
index: 2; visibility: inherit; width: 100%; height:100%"></iframe>
      </td>
    </tr>
</table>
</div>
</body>
</html>
<%@ page import="java.text.*"%>
<%@ page import="java.sql.*"%>
<jsp:useBean id="workD" scope="page" class="pub.connonline" />
<jsp:useBean id="workT" scope="page" class="pub.mytime"/>
<%!
String userid="",truename="",nickname="",userrole="";
String mycommunity="",myroom="";
String note="";
String sqlString="";
String mytime="";
%>
<%
mycommunity=,request.getParameter("topic_id_1").trim();
myroom=,request.getParameter("topic_id_2").trim();
%>
<%
userrole=(String)session.getAttribute("userrole").trim();
if("学生".equals(userrole))
{
     nickname=(String)session.getAttribute("nickname");
     userid=(String)session.getAttribute("studentid");
```

```
}
else if("管理员".equals(userrole))
{
     nickname = (String)session.getAttribute("nickname");
     userid = (String)session.getAttribute("adminid");
}
%>
<%
note = "『系统通知』: " + accountname + "加入了我们的讨论";
mytime = workT.getTime();
//将系统通知插入到聊天记录数据表 chat 中,mytime1 记录聊天时间到小时。
sqlString = "insert into chat(topic_id_1,topic_id_2,username,userid,userclass,to_user,
color,face,note,addtime,addtime1,mode)" + " values('" + mycommunity + "','" + myroom + "','" +
accountsname + "','" + userid + "','" + userrole + "','大家','#000000','无','" + note + "','" + mytime
+ "','" + mytime.substring(0,13) + "','1')";
try
{
    workD.executeUpdate(sqlString);
}
catch(Exception ex)
{
    out.print(ex.getMessage());
}
  finally
  {
      workD.disconnectToDB();
  }
%>
<%
//每隔一小时从数据表 chat 中清除聊天记录,addtime1 记录聊天时间到小时
mytime = mytime.substring(0,13);
String sqlString = "delete from chat where addtime1! = '" + mytime + "'";
try
{
    workD.executeUpdate(sqlString);
}
catch(Exception ex)
{
    out.print(ex.getMessage());
}
finally
{
    workD.disconnectToDB();
}
%>
```

在上述代码中,为了保证界面的灵活性,引用了一个 CSS 文件 chat.css,代码从略。

### 2. 左侧信息列表 chat_left.jsp

左侧显示用户列表,代码如下:

```
<%@ page contentType="text/html;charset=GBK" %>
<%@ page import="java.text.*" %>
<%@ page import="java.sql.*" %>
<jsp:useBean id="workD" scope="page" class="pub.connonline" />
<%!
String userid="";
String truename="",nickname="";
String userrole="";
String sqlString="";
String topic_id_1="";
String topic_id_2="";
String topic_name="";
ResultSet rs = null;
%>
<%
userrole=(String)session.getAttribute("userrole").trim();

if("学生".equals(userrole))
{
     nickname=(String)session.getAttribute("nickname");
     userid=(String)session.getAttribute("studentid");
}
else if("管理员".equals(userrole))
{
     nickname=(String)session.getAttribute("nickname");
     userid=(String)session.getAttribute("admin_name");
}
topic_id_1=(String)session.getAttribute("topic_id_1");
topic_id_2=(String)session.getAttribute("topic_id_2");
%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<link rel="stylesheet" type="text/css" href="scripts/left.css">
<script language="JavaScript">
 function PerformSubmit(user)
{
       parent.bottom.form1.to_user.value=user;
       return false;
  }
</SCRIPT>
</head>
<body>
<table class="table" cellpadding="0" cellspacing="0" style="margin-top:10px;">
<tr><td class="row1"><b>GSL | 在线交流</b></td></tr>
<tr>
<td class="row2">
<table class="welcome" cellpadding="0" cellspacing="0" border="0">
        <tr><td height="20px"><b>欢迎来到在线交流社区</b></td></tr>
```

```
            <tr><td height = "15px"></td></tr>
            <tr><td height = "20px">现在所在主题：</td></tr>
            <tr><td height = "5px"></td></tr>
<%
try
{
    rs = workD.executeQuery("select * from second_topic where id = '" + topic_id_2.trim() + "'");
    rs.next();
    topic_name = rs.getString("topic_name");
}
catch(Exception ex)
{
    out.print(ex.getMessage());
}
finally
{
    rs = null;
    workD.disconnectToDB();
}
%>
    <tr><td height = "20px"><font color = "#0033FF"><% = topic_name %></font>
</td></tr>
    </table>
</td>
</tr>
</table>
<table class = "table" cellpadding = "0" cellspacing = "0" style = "margin-top:15px;">
<tr><td class = "row1"><b>个人信息</b></td></tr>
<tr>
 <td class = "row4">
<table id = "welcome2" cellpadding = "0" cellspacing = "0" border = "0">
 <tr><td height = "20px">用户账号：<font color = "#0033FF"><% = userid %>
</font></td> </tr>
     <tr><td height = "5px"></td></tr>
     <tr> <td height = "20px">姓名：<font color = "#0033FF"><% = nickname %>
</font></td></tr>
     <tr><td height = "5px"></td></tr>
     <tr><td height = "20px">角色：<font color = "#0033FF"><% = userrole %>
</font></td></tr>
</table>
</td>
</tr>
</table>
<table class = "table" cellpadding = "0" cellspacing = "0" style = "margin-top:15px;">
<tr><td class = "row1"><b>在线信息</b></td></tr>
<tr>
 <td class = "row3">
<table id = "welcome1" cellpadding = "0" cellspacing = "0" border = "0">
<%
sqlString = "select username from chat where topic_id_1 = '" + topic_id_1.trim() + "' and topic_
id_2 = '" + topic_id_2.trim() + "' group by username";
```

```
rs = workD.executeQuery(sqlString);
rs.last();
int online_nums = rs.getRow();
%>
  <tr><td height = "20px">当前在线 <% = online_nums %>人</td></tr>
  <tr><td height = "5px"></td></tr>
  <tr><td height = "20px">在线用户列表：</td></tr>
<%
rs.absolute(1);
do
{
%>
  <tr>
     <td height = "20px"><a href = "#" onClick = 'PerformSubmit("<% = rs.getString
("username")%>")'><% = rs.getString("username")%></a></td>
  </tr>
<%
}
while(rs.next());
%>
   <tr><td> </td></tr>
 </table>
</td>
</tr>
</table>
<table class = "table" cellpadding = "0" cellspacing = "0" style = "margin-top:15px;">
<tr><td class = "row1"><b><a href = "exit.jsp" target = '_parent'>离开该话题</a></
b></td></tr>
</table>
</body>
</html>
```

需要说明的是，当一个用户登录聊天界面时，系统发一个公告消息，同时将该用户存储在聊天数据表 chat 中，通过该数据表，可以显示在线用户列表。

### 3. 用户输入页面 chat_input.jsp

在聊天程序界面的右下部，是用户的输入区，输入信息后，同时将输入信息保存到聊天数据库中。在 chat_input.jsp 文件中，Form 表单的 action 属性设置为 chat_input.jsp 文件本身，第一次连接到该文件时，即进入聊天室，只将公告存储到聊天数据表 chat 中。

```
<%@ page contentType = "text/html;charset = GBK" %>
<%@ include file = "../session-guard.jsp" %>
<%@ page import = "java.text.*" %>
<%@ page import = "java.sql.*" %>
<jsp:useBean id = "workD" scope = "page" class = "pub.connonline" />
<jsp:useBean id = "workT" scope = "page" class = "pub.mytime"/>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns = "http://www.w3.org/1999/xhtml">
```

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<link rel="stylesheet" type="text/css" href="scripts/bottom.css">
<script language="javascript">
 function on_submit()
 {  //验证数据的合法性
    if (form1.content.value =="")
    {
      alert("请输入内容!");
      form1.content.focus();
      return false;
    }
 }
</script>
</head>
<body>
<form action="chat_input.jsp" method="post" onSubmit="return on_submit()" name="form1">
<table style="width:600px; height:100px; margin-top:15px; font-size:12px; text-align:
center; vertical-align:middle;" cellpadding="0" cellspacing="0" border="0">
<tr>
 <td height="30px" style="text-align:center; margin-right:30px;">
 对象:<input class="pas" type="text" name="to_user" value="大家"/>  字体
颜色:
<select name='fontcolor' id='fontcolor' size="1" style="font-size:9pt">
<OPTION value='#3333ff' style='color:#3333ff'>默认</option>
<option value='#000000' style='color:#000000'>黑色</option>
<OPTION value='#ff0000' style='color:#ff0000'>大红</option>
<OPTION value='#0088ff' style='color:#0088ff'>海蓝</option>
<OPTION value='#ff0088' style='color:#ff0088'>玫瑰</option>
<OPTION value='#ff8800' style='color:#ff8800'>金黄</option>
</option>
</SELECT>  
 动作与表情:<SELECT name='mode' id='mode'>
<OPTION value='微笑着'>微笑</option>
<OPTION value='摇头'>摇头</option>
<OPTION value='点头'>点头</option>
<OPTION value='惊呼'>惊呼</option>
<OPTION value='思忖着'>思忖</option>
<OPTION value='高兴地'>高兴</option>
<OPTION value='满意地'>满意</option>
<OPTION value='痛苦地'>痛苦</option>
</SELECT>
   <font color="#0033FF">留言内容不要超过150字</font>
</td>
</tr>
<tr>
 <td height="60px">内容:<textarea name="content" cols="70" rows="3" class=
"textarea"></textarea></td>
</tr>
<tr>
 <td height="20px"><input type="submit" value="提交" class="menu">  
```

```
<input type="reset" value="重置" class="menu"></td>
</tr>
</table>
</form>
</body>
</html>
<%!
String truename="",nickname="";
String userole="";
String userid="";
String sqlString="";
String topic_id_1="";
String topic_id_2="";
String time="";
String chat_record="";
%>
<%!
public String codeToString(String str)
{  //处理中文字符串的函数
    String s=str;
    try
      {
        byte tempB[]=s.getBytes("ISO-8859-1");
        s=new String(tempB);
        return s;
      }
     catch(Exception e)
     {
       return s;
      }
}
%>
<%//接收客户端提交的数据
 String to_user=codeToString(request.getParameter("to_user"));
 if (to_user==null)       // 无内容则设为空串
     to_user="大家";
 String fontcolor=codeToString(request.getParameter("fontcolor"));
 if (fontcolor==null)     //无内容则设为空串
     fontcolor="";
 String mode=codeToString(request.getParameter("mode"));
 if (mode==null)          //无内容则设为空串
     mode="";
 String content=codeToString(request.getParameter("content"));
 if(content==null)        //无内容则设为空串
     content="";
%>
<%
userrole=(String)session.getAttribute("userrole").trim();
if("学生".equals(userrole))
{
     nickname=(String)session.getAttribute("nickname");
```

```
        userid = (String)session.getAttribute("studentid");
    }
    else if("管理员".equals(userrole))
    {
        nickname = (String)session.getAttribute("nickname");
        userid = (String)session.getAttribute("admin_name");
    }
    topic_id_1 = (String)session.getAttribute("topic_id_1");
    topic_id_2 = (String)session.getAttribute("topic_id_2");
    %>
    <%
    if(!("".equals(content))) //如果输入内容不为空,则执行插入数据库操作
    {
        chat_record = accountsname + mode + "对" + to_user + "说：" + content;
        time = workT.getTime();
        sqlString = "insert into chat(topic_id_1,topic_id_2,username,userid,userclass,to_user,
    color,face,note,addtime,addtime1,mode)" + " values('" + topic_id_1 + "','" + topic_id_2 + "',
    '" + accountsname + "','" + author_id + "','" + user_class + "','" + to_user + "','" + fontcolor +
    "','" + mode + "','" + chat_record + "','" + time + "','" + time.substring(0,13) + "','5')";
        try
        {
            workD.executeUpdate(sqlString);
        }
        catch(Exception ex)
        {
            out.print(ex.getMessage());
        }
        finally
        {
            workD.disconnectToDB();
        }
    }
    %>
```

上述代码分为两个部分,第一部分是显示界面,第二部分的 JSP 程序用于存储聊天记录。若聊天内容输入为空,则不执行数据表插入。

#### 4. 聊天信息显示 chat_client.html

在聊天程序界面中,右上侧显示聊天信息,为避免刷新整个页面而引起的屏幕闪烁,采用 AJAX 技术,客户端代码为一个 HTML 页面 chat_client.html,代码清单如下:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv = "Content-Type" content = "text/html; charset = GBK">
<link rel = "stylesheet" type = "text/css" href = "scripts/mainframe.css">
</head>
<body onload = "sendRequest()">
<div style = " width:650px; height:400px; overflow:scroll; margin-top:20px; background:
```

```
#FFFFFF; margin-left:10px;">
<div id="chat_content" style="width:610px; margin-top:5px; margin-left:5px; text-align:left; line-height:150%;">
</div>
</div>
<script>
// 在 Tomcat 中,调试时输出产生换行,在 text 框中找不出原因,通过输出到 DIV,发现以
// 下字符,使用正则表达式将其替换为空即可
function convert(str)
{
      if (str!="")
      {
        str=str.replace(/\r/g,"");
      }
      return str;
}
var XMLHttpReq;
// 创建 XMLHttpRequest 对象
function createXMLHttpRequest()
{
     if(window.XMLHttpRequest)
     {  //Mozilla 浏览器
       XMLHttpReq = new XMLHttpRequest();
     }
     else
     if (window.ActiveXObject)
     {  // IE 浏览器
       try
       {
         XMLHttpReq = new ActiveXObject("Msxml2.XMLHTTP");
       }
       catch (e)
       {
         try
         {
           XMLHttpReq = new ActiveXObject("Microsoft.XMLHTTP");
         }
         catch (e)
         {
         }
     }
   }
}
function sendRequest()
{
     createXMLHttpRequest();
     var url = "chat_server.jsp";                          //设置服务端通信程序
     XMLHttpReq.open("POST", url, true);
     XMLHttpReq.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
     XMLHttpReq.onreadystatechange = processResponse;      //指定响应函数
     XMLHttpReq.send();                                    // 发送请求
```

```
        setTimeout("sendRequest()" , 800);
    }
    // 处理返回信息函数
    function processResponse()
    {
          if (XMLHttpReq.readyState ==4)
          {  // 判断对象状态
            if (XMLHttpReq.status ==200)
            {  // 信息已经成功返回,开始处理信息,在 div 中显示聊天记录
               chat_content.innerHTML = convert(XMLHttpReq.responseText);
            }
            else
            { //页面不正常
                window.alert("您所请求的页面有异常。");
              }
            }
      }
    </script>
    </body>
    </html>
```

现在,Web 开发人员更多地使用 div 和 css 来进行页面布局,div 可以起一个占位作用(定义图层的位置、高度和宽度),然后通过程序来动态地修改 div 中的内容,或者改变 div 的显示状态。在上述程序中,使用图层(id="chat_content")来定义聊天记录滚动显示区域。用户也可以通过 textarea 来定义显示区域,两者的不同是在 div 定义中,可以格式化显示 HTML 文本,但 textarea 中,不能显示 HTML 格式化内容。

### 6.8.3 服务端设计

在服务端,主要完成聊天记录的数据存储和查找操作。主要涉及的程序就是用户输入中的服务端脚本部分,同时还涉及一个与客户端聊天信息显示进行异步传输的服务端程序,两者共同构成 AJAX 的异步传输,来避免用户界面的闪烁。

#### 1. 数据库结构

聊天数据库记录聊天社区、聊天主题和聊天记录,包括三个数据表:社区数据表、主题数据表(房间)和聊天记录数据表,结构定义如下。

(1) 社区数据表 topic

社区数据表存储要进入的一级主题,结构定义如图 6-16 所示。

(2) 主题数据表(房间)second_topic

主题数据表(房间)second_topic,存储聊天主题,即一般的房间,结构定义如图 6-17 所示。

(3) 聊天记录数据表 chat

记录聊天信息以及系统公告,数据结构定义如图 6-18 所示。

表 - dbo.topic 摘要

| 列名 | 数据类型 | 允许空 |
| --- | --- | --- |
| id | int | ☐ |
| topic_name | nvarchar(255) | ☑ |
| topic_note | nvarchar(255) | ☑ |
| key_word | nvarchar(255) | ☑ |
| | | ☐ |

图 6-16 社区数据表 topic 结构定义

| 列名 | 数据类型 | 允许空 |
| --- | --- | --- |
| id | int | ☐ |
| topic_id_1 | nvarchar(255) | ☑ |
| topic_name | nvarchar(255) | ☑ |
| topic_note | ntext | ☑ |
| | | ☐ |

表 - dbo.second_topic　摘要

图 6-17　主题数据表(房间)second_topic 结构定义

| 列名 | 数据类型 | 允许空 |
| --- | --- | --- |
| id | int | ☐ |
| topic_id_1 | char(10) | ☑ |
| topic_id_2 | char(10) | ☑ |
| username | nvarchar(50) | ☑ |
| userid | nvarchar(50) | ☑ |
| userclass | nvarchar(50) | ☑ |
| to_user | nvarchar(50) | ☑ |
| color | nvarchar(50) | ☑ |
| face | char(10) | ☑ |
| note | nvarchar(MAX) | ☑ |
| addtime | nvarchar(50) | ☑ |
| addtime1 | nvarchar(50) | ☑ |
| mode | char(10) | ☑ |
| | | ☐ |

表 - dbo.chat　摘要

图 6-18　主题数据表(房间)second_topic 结构定义

### 2. 聊天信息显示对应的服务端程序 chat_server.jsp

根据 Ajax 技术的工作机理，对应客户端的 chat_client.html 文件，服务器端的处理程序为 chat_server.jsp，代码清单如下：

```
<%@ page contentType="text/html;charset=GBK" %>
<%@ include file="../session-guard.jsp" %>
<%@ page import="java.text.*" %>
<%@ page import="java.sql.*" %>
<jsp:useBean id="workD" scope="page" class="pub.connonline" />
<%!
ResultSet rs = null;
String topic_id_1="";
String topic_id_2="";
String color_text="";
String color_time="#999999";
String chat_content="";
String chat_time="";
%>
<%!
public String codeToString(String str)
{ //处理中文字符串的函数
  String s=str;
  try
  {
    byte tempB[]=s.getBytes("ISO-8859-1");
    s=new String(tempB);
    return s;
  }
  catch(Exception e)
  {
    return s;
  }
}
```

```
%>
<%
topic_id_1 = (String)session.getAttribute("topic_id_1");
topic_id_2 = (String)session.getAttribute("topic_id_2");
%>
<%
try
{   //topic_id_1 为一级主题 id,topic_id_2 为二级主题 id,
    // topic_id_1 = '0'且 topic_id_2 = '0'为系统公告
    rs =  workD.executeQuery("select  *  from chat where (topic_id_1 = '" + topic_id_1.trim() + "' and
                topic_id_2 = '" + topic_id_2.trim() + "') or (topic_id_1 = '0' and topic_id_2 =
'0') order by id");
    while(rs.next())
    {
        color_text = rs.getString("color");
        chat_content = rs.getString("note");
        chat_time = rs.getString("addtime");
        out.println ("<font color = " + color_text + ">" + chat_content + "</font> " + "<font
                color = " + color_time + ">(" + chat_time + ")</font>");
        out.println("<br>");
    }
}//外层 try 结束
catch(Exception ex)
{
    out.print(ex.getMessage());
}
finally
{
    rs = null;
    workD.disconnectToDB();
}
%>
```

### 3. JavaBeans 类

为了更好地实现 MVC 设计模式，对于数据库操作中的数据库连接，设计成一个 Java 类 connonline.java，存储在系统根目录下的\WEB-INF\classes\pub 包中，此外还包含一个时间操作的类 mytime.java。

(1) 连接数据库 connonline.java

```
package pub;
import java.sql.*;
public class connonline
{
  String sConnStr = "jdbc:sqlserver://localhost:1433;DatabaseName = gsl_myonline";
  String sDBDriver = "com.microsoft.sqlserver.jdbc.SQLServerDriver";
  String strUser  =  "sa";
  String strPassword  =  "sa";
  Connection conn  =  null;
  public ResultSet rs  =  null;
```

```
  private java.sql.Statement stmt = null;
  public connonline()
  {
  try
  {
    Class.forName(sDBDriver).newInstance();
  }
  catch(Exception e)
  {
      System.err.println("connonline(): " + e.getMessage());
     }
  }
  public ResultSet executeQuery(String sql)
  {
    rs = null;
    try
     {
      conn = DriverManager.getConnection(sConnStr, strUser, strPassword);
      //Statement stmt = conn.createStatement();
      Statement stmt = conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
                                            ResultSet.CONCUR_READ_ONLY);
      rs = stmt.executeQuery(sql);
     }
  catch(SQLException ex)
   {
      System.err.println("aq.executeQuery: " + ex.getMessage());
   }
   return rs;
}
public int executeUpdate(String sql)
{
    int returnVal = -999;
    try
    {
    conn = DriverManager.getConnection(sConnStr, strUser, strPassword);
    Statement stmt = conn.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                                          ResultSet.CONCUR_UPDATABLE);
    returnVal = stmt.executeUpdate(sql);
   }
  catch(SQLException ex)
  {
      System.err.println("aq.executeUpdate: " + ex.getMessage());
   }
   return returnVal;
  }
  public void disconnectToDB() throws java.sql.SQLException
  {
      if (rs!=null)
      {
          rs.close();
          rs = null;
```

```
            }
            if (stmt! = null)
            {
                stmt.close();
                stmt = null;
            }
            if (conn! = null)
            {
                conn.close();
                conn = null;
            }
        }
    }
```

(2) 获取当前时间 mytime.java

在聊天系统中,时间的应用很多,在 Java 中,时间的处理涉及的类较多,我们可以编写一个自己的类 mytime,来处理时间,代码清单如下:

```
package pub;
public class mytime {
   public String getTime()
    {
      String datestr = "" ;
      try
      {
        java.text.DateFormat df = new java.text.SimpleDateFormat("yyyy - MM - dd HH:mm") ;
        java.util.Date date = new java.util.Date() ;
        datestr = df.format(new java.util.Date()) ;
      }
      catch (Exception ex)
      {
      }
      return datestr ;
     }
     public String getcurrenttime()
      {
        String curtime = "";
        try
        {
          java.text.DateFormat jd = new java.text.SimpleDateFormat("HH:mm:ss");
          java.util.Date ud = new java.util.Date();
          curtime = jd.format(ud);
         }
        catch(Exception ex){
        }
     return curtime;
  }
}
```

#### 4. 会话安全程序 session-guard.jsp

为了防止用户直接通过 URL 进入一个网页,在每一页的开始,可以执行一个会话检查

程序，如果用户不是通过 login 登录界面正常进入页面，将强行把连接导航到相应的页面。session-guard.jsp 代码清单如下：

```
<%@ page pageEncoding="GBK" %>
<%
  //从 login 正常登录时，设置会话属性值
  //设置 session.setAttribute("UNIQUE_CHECK_STRING", username);
  String uniqueCheckString = (String)session.getAttribute("UNIQUE_CHECK_STRING");
  if (uniqueCheckString ==null)
    { //销毁当前 Session
     session.invalidate();
%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>系统错误提示</title>
</head>
<body style="margin:0 auto; font-size:12px; text-align:center; background:#FDF6EE; ">
<table cellpadding="0" cellspacing="0" border="0" style="width:400px; margin-top:
150px; text-align:left; vertical-align:top; border-left:1px solid #0163A2; border-top:
1px solid #0163A2;">
<tr>
  <td style=" border-bottom:1px solid #0163A2;"><img src="images/err.jpg" width=
"70" height="70" /></td>
  <td style="text-align:left; vertical-align:middle; text-indent:1em; line-height:
150%; border-bottom:1px solid #0163A2; border-right:1px solid #0163A2;">您的页面已经
过期或者您未从登录页面登录！<br>
  请重新登录……</td>
</tr>
<tr>
  <td style="text-align:center; vertical-align:middle; " colspan="2">
    <a href="index.jsp" target="_self">返回重新登录</a>
  </td>
</tr>
</table>
</body>
</html>
<%
  }
%>
```

上述 session_gard.jsp 代码通过<%@ include file="../session-guard.jsp" %>语句被包含在各个网页的开始，以确保用户必须通过账户正常的登录来访问网页，避免直接通过某个页面的网址进入网页。

## 6.9 Java 开发工具简介

在计算机开发语言的历史中，从来没有哪种语言像 Java 那样受到如此众多厂商的支持，有如此多的开发工具。每一种 Java 开发工具都各有所长，各有特点。下面对一些常见

的 Java 开发工具进行简要介绍。

### 6.9.1 JDK(Java Development Kit)

Java 开发工具包(Java Development Kit)JDK 是整个 Java 的核心,包括 Java 运行环境(Java Runtime Envirnment),Java 基础类库和一组建立、测试及建立文档的 Java 实用程序,这些实用程序包括:开发用的 Java 编译器(javac)、Java 解释器(java)、打包工具(jar)、文档生成器(javadoc)、调试工具(jdb)等。不论什么 Java 应用服务器实质都是内置了某个版本的 JDK,掌握 JDK 是学好 Java 的第一步。

最主流的 JDK 是 Sun 公司发布的 JDK,一般有三种版本,即:J2SE(标准版)、J2EE(企业版)和 J2ME(Micro Edtion),其中,J2SE 为一般用户常用的开发环境,J2EE 主要用于企业级 J2EE 应用程序,而 J2ME 则用于移动设备、嵌入式设备上的 Java 应用程序开发。不同的版本包含的包也不相同。除了 Sun 之外,还有很多公司和组织都开发了自己的 JDK,例如 IBM 公司开发的 JDK,BEA 公司的 Jrocket,还有 GNU 组织开发的 JDK 等等。其中 IBM 的 JDK 包含的 JVM(Java Virtual Machine)运行效率要比 Sun JDK 包含的 JVM 高出许多。而专门运行在 x86 平台的 Jrocket 在服务端运行效率也要比 Sun JDK 好很多。

JDK 简单易学,可以通过任何文本编辑器(如:Windows 记事本、UltrEdit、Editplus、FrontPage 以及 DreamWeaver 等)编写 Java 源文件,然后在 DOS 状态下通过 javac 命令将 Java 源程序编译成字节码(.class 文件),通过 Java 命令来执行编译后的 Java 文件,这能带给 DOS 时代程序员美好的回忆。

从初学者角度来看,采用 JDK 开发 Java 程序能够很快理解程序中各部分代码之间的关系,有利于理解 Java 面向对象的设计思想。但它的缺点也是非常明显的,就是从事大规模企业级 Java 应用开发非常困难,不能进行复杂的 Java 软件开发,也不利于团体协同开发。即便如此,JDK 仍然是许多 Java 专家最初使用的开发环境,尽管许多编程人员已经使用第三方的开发工具,但 JDK 仍被当作 Java 开发的重要工具。

### 6.9.2 Sun NetBeans 集成开发环境

NetBeans 是一个全功能的开放源码 Java IDE,可以帮助开发人员编写、编译、调试和部署 Java 应用,并将版本控制和 XML 编辑融入其众多功能之中。NetBeans 可支持 Java 2 平台标准版(J2SE)应用的创建、采用 JSP 和 Servlet 的二层 Web 应用的创建,以及用于二层 Web 应用的 API 及软件的核心组的创建。此外,NetBeans 还预装了一个 Web 服务器,即 Tomcat,从而免除了繁琐的配置和安装过程。所有这些都为 Java 开发人员创造了一个可扩展的开源多平台的 Java IDE,以支持他们在各自所选择的环境中从事开发工作,如 Solaris、Linux、Windows 或 Macintosh。

在 NetBeans 中,不仅可以开发 Java 程序,通过安装插件,也可以开发 C/C++程序,目前的版本是 NetBeans IDE 6.1,其中 NetBeans IDE 3 和 NetBeans IDE 5.0 都有中文版本,是目前进行大型 Web 应用开发的常用开发环境。用户可以从 Sun 的官方网站下载,网址是:http://www.sun.com/。

### 6.9.3 JBuilder开发环境

有人说Borland的开发工具都是里程碑式的产品，从Turbo C、Turbo Pascal到Delphi、C++ Builder都是经典。JBuilder是Borland公司开发的针对Java的开发工具，使用JBuilder将可以快速、有效地开发各类Java应用，它使用的JDK与Sun公司标准的JDK不同，它经过了较多的修改，以便开发人员能够像开发Delphi应用那样开发Java应用。

下面简单介绍一下JBuilder的特点：

(1) JBuilder支持最新的Java技术，包括Applets、JSP/Servlets、JavaBean以及EJB(Enterprise JavaBeans)的应用。

(2) 用户可以自动地生成基于后端数据库表的EJB Java类，JBuilder同时还简化了EJB的自动部署功能。此外它还支持CORBA，相应的向导程序有助于用户全面地管理IDL(分布应用程序所必需的接口定义语言Interface Definition Language)和控制远程对象。

(3) JBuilder支持各种应用服务器。JBuilder与Inprise Application Server紧密集成，同时支持WebLogic Server，支持EJB 1.1和EJB 2.0，可以快速开发J2EE的电子商务应用。

(4) JBuilder能用Servlet和JSP开发和调试动态Web应用。

(5) 利用JBuilder可创建(没有专有代码和标记)纯Java2应用。由于JBuilder是用纯Java语言编写的，其代码不含任何专属代码和标记，它支持最新的Java标准。

(6) JBuilder拥有专业化的图形调试界面，支持远程调试和多线程调试，调试器支持各种JDK版本，包括J2ME/J2SE/J2EE。

JBuilder环境开发程序方便，它是纯的Java开发环境，适合企业的J2EE开发；缺点是往往一开始难以把握整个程序各部分之间的关系，对机器硬件要求较高，运行速度较慢。

### 6.9.4 Eclipse开发平台

2001年11月，IBM公司捐出价值4000万美元的源代码组建了Eclipse联盟，业界厂商合作创建了Eclipse平台。Eclipse允许在同一IDE中集成来自不同供应商的工具，并实现了工具之间的互操作性。从本质上说，Eclipse是一个开放源代码的、基于Java的可扩展开发平台，它只是一个框架和一组服务，用于通过插件组件构建开发环境，它附带了一个标准的插件集，包括Java开发工具(Java Development Tools，JDT)。

Eclipse是一个Java IDE，但Eclipse的目标不仅限于此。Eclipse还包括插件开发环境(Plug-in Development Environment，PDE)，这个组件主要针对希望扩展Eclipse的软件开发人员，因为它允许他们构建与Eclipse环境无缝集成的工具。Eclipse是使用Java语言开发的，但它的用途并不限于Java语言；例如，支持诸如C/C++、COBOL和Eiffel等编程语言插件，Eclipse框架还可用来作为与软件开发无关的其他应用程序类型的基础，比如内容管理系统。

基于Eclipse的应用程序的突出例子是IBM的WebSphere Studio Workbench，它构成

了 IBM Java 开发工具系列的基础。例如,WebSphere Studio Application Developer 添加了对 JSP、servlet、EJB、XML、Web 服务和数据库访问的支持。Eclipse 是一款非常受欢迎的 Java 开发工具,用户越来越多。但是,它的缺点就是较复杂,对初学者来说,理解起来比较困难。

### 6.9.5 JDeveloper 开发框架

在 Web 开发中,使用 JSP(Java Server Page)和 EJB(Enterprise JavaBeans)进行 J2EE(Java 2 Enterprise Edition)开发是很普遍的。但是,J2EE 相当繁杂,为了部署高效的 J2EE 应用程序,必须掌握许多知识。为了简化开发者的工作,Oracle 提出了应用开发框架(Application Development Framework ,ADF)的概念,Oracle ADF 的目标是降低繁杂度,提高应用程序的整体效能。

在 Oracle ADF 下,Oracle JDeveloper 为构建具有 J2EE 功能、XML 和 Web services 的复杂的、多层的 Java 应用程序提供了一个完全集成的开发环境。它为运用 Oracle 数据库和应用服务器的开发人员提供特殊的功能和增强性能,除此以外,它也成为用于多种用途 Java 开发的一个强大的开发工具。

Oracle9i JDeveloper 的主要特点如下:

(1) 具有 UML(Unified Modeling Language,一体化建模语言)建模功能。可以将业务对象及 e-business 应用模型化。

(2) 配备有高速 Java 调试器(Debuger)、内置 Profiling 工具、提高代码质量的工具"CodeCoach"等。

(3) 支持简单对象访问协议(Simple Object Access Protocol,SOAP)、统一描述、发现和集成协议(Universal Description, Discovery and Integration,UDDI)、Web 服务描述语言(Web Services Description Language,WSDL)等 Web 服务标准。

总之,Oracle JDeveloper 提供了业界一个完整的、集成了 J2EE 和 XML 的开发环境,允许开发者快速开发可以通过 Web、无线设备及语音界面访问的 Web 服务和交易应用,以往只能通过将传统 Java 编程技巧与最新模块化方式结合到一个单一集成的开发环境中之后才能完成 J2EE 应用开发生命周期管理的事实,从根本上得到改变。缺点就是对于初学者来说,较复杂,也比较难。

### 6.9.6 其他工具和资源

除了上述一些常见的工具外,常用的 Java 开发工具还有:

Visual Café for Java(Symantec 公司)、Visual Age for Java(IBM)、Sun Java Studio 5(Sun)、Java Workshop(Sun)、WebLogic Workshop(BEA 公司)、JRUN(Macromedia 公司)、JCreator(Sun)、Microsoft Visual J++(微软)、雅加达蚂蚁 ANT(Apache 开放源码组织)、IntelliJ IDEA(IntelliJ 公司)等。

另外,有关 Java 的其他资源还有:

CSDN 论坛(http://www.csdn.net/)是中国最有名的技术论坛,也是学习和交流 Java

技术非常有名的场所,《程序员》杂志就是他们出版的。

Java 研究组织 http://www.javaresearch.org/ ,有很多原创文章,不乏许多高手的文章。

Java 中文世界论坛 http://www.chinajavaworld.com/ ,有比较全面的 Java 资料。

另外,可以通过 www.google.com 找到众多关于 Java 技术和开发的文章,也是大家学习 Java、进行 Web 开发必不可少的手段。

## 思　考　题

1. 简述 Java 的技术特征。
2. 为什么说 Java 是一种完全面向对象的程序设计语言?
3. 简述对象和类的概念。
4. 什么是多态? 多态有哪些实现形式?
5. 什么是接口? 说明接口和抽象类的关系。
6. 什么是 JDK? 都有哪些公司提供 JDK 产品? 他们各有什么特点?
7. 列举主要的 Java 开发环境,它们的 JDK 相同吗?
8. 说明 Web 应用中的三层体系结构,并说明它的优势。
9. 什么是 JavaApplet? JavaApplet 的主要应用是什么? 它和 Java 应用程序有何不同?
10. 什么是 JavaBeans? JavaBeans 和 Java 类有何不同?
11. 什么是 CGI? 说明 CGI 的主要功能。
12. 什么是 Servlet? 它和普通的 Java 类有何不同? 如何运行一个 Servlet?
13. 在 JSP 中用 Bean 和 Servlet 联合实现用户注册、登录功能。
14. 在 JSP 内置对象中,简述 request 对象的功能。
15. 什么是会话(session)? session 对象的主要用途是什么?
16. 使用 JSP 制作留言板。
17. 使用 JSP,编写购物程序。
18. 使用 JSP,根据一家商场的销售情况,生成饼图。
19. 进一步完善 6.8 节的在线聊天程序,例如增加私聊等功能。

# 参 考 文 献


1. 吴鹤龄,崔林. ACM 图灵奖——计算机发展史的缩影. 北京:高等教育出版社,2002
2. 曾明. World Wide Web 开发使用指南. 北京:人民邮电出版社,1996
3. [美]Douglas E. Comer 著. Internet 导引. 马志强,廖卫东译. 北京:清华大学出版社,1995
4. 金勇华,曲俊生. Java 网络高级编程. 北京:人民邮电出版社,2002
5. 孙卫琴,李洪成. Tomcat 与 Java Web 开发技术详解. 北京:电子工业出版社,2004
6. 孙卫琴. Java 面向对象编程. 北京:电子工业出版社,2006
7. 夏先波. Java JDK 实例宝典. 北京:电子工业出版社,2007
8. 林信良. Java JDK 6 学习笔记. 北京:清华大学出版社,2007
9. 王国辉,王毅,尹相群. Java Web 开发技术方案宝典. 北京:人民邮电出版社,2008
10. 袁建洲. JavaScript 编程宝典. 北京:电子工业出版社,2006
11. 曹衍龙,叶达峰. AJAX 编程技术与实例. 北京:人民邮电出版社,2007
12. 刘乃丽. 精通 Java EE 项目开发——基于 Eclipse、Spring、Struts、Hibernate. 北京:人民邮电出版社,2008
13. 彭晖,史忠植. 语义 Web:让计算机读懂互联网. 计算机世界报,第 45 期,2007. 11. 26
14. Uche Ogbuji. Thinking XML:XML 十年发展历程. http://www. ibm. com/developerworks/cn/xml/x-think38. html
15. Java 开发技术十年的回顾与展望. 编程中国. http://www. bccn. net/Article/kfyy/java/
16. 郝兴伟. Web 开发技术. 北京:高等教育出版社,2007
17. 中国 XML 论坛. http://www. xml. org. cn/index. asp
18. Java 中文世界论坛. http://bbs. chinajavaworld. com/index. jspa
19. SOA 中国技术论坛. http://soachinaforum. com/

# 读者意见反馈

亲爱的读者：

感谢您一直以来对清华版计算机教材的支持和爱护。为了今后为您提供更优秀的教材，请您抽出宝贵的时间来填写下面的意见反馈表，以便我们更好地对本教材做进一步改进。同时如果您在使用本教材的过程中遇到了什么问题，或者有什么好的建议，也请您来信告诉我们。

地址：北京市海淀区双清路学研大厦 A 座 602 室　计算机与信息分社营销室　收
邮编：100084　　电子邮件：jsjjc@tup.tsinghua.edu.cn
电话：010-62770175-4608/4409　　邮购电话：010-62786544

教材名称：Web 技术导论（第 2 版）

ISBN 978-7-302-19371-5

**个人资料**

姓名：________ 年龄：______所在院校/专业：________

文化程度：________ 通信地址：________

联系电话：________ 电子信箱：________

**您使用本书是作为：**□指定教材 □选用教材 □辅导教材 □自学教材

**您对本书封面设计的满意度：**

□很满意 □满意 □一般 □不满意　改进建议________

**您对本书印刷质量的满意度：**

□很满意 □满意 □一般 □不满意　改进建议________

**您对本书的总体满意度：**

从语言质量角度看　□很满意 □满意 □一般 □不满意

从科技含量角度看　□很满意 □满意 □一般 □不满意

**本书最令您满意的是：**

□指导明确 □内容充实 □讲解详尽 □实例丰富

**您认为本书在哪些地方应进行修改？（可附页）**

________

________

**您希望本书在哪些方面进行改进？（可附页）**

________

________

# 电子教案支持

敬爱的教师：

为了配合本课程的教学需要，本教材配有配套的电子教案（素材），有需求的教师可以与我们联系，我们将向使用本教材进行教学的教师免费赠送电子教案（素材），希望有助于教学活动的开展。相关信息请拨打电话 010-62776969 或发送电子邮件至 jsjjc@tup.tsinghua.edu.cn 咨询，也可以到清华大学出版社主页（http://www.tup.com.cn 或 http://www.tup.tsinghua.edu.cn）上查询。

# 高等学校教材·计算机应用
## 系列书目

| 书　　号 | 书　　名 | 作　　者 |
|---|---|---|
| 9787302081432 | C++语言程序设计教程与实验 | 温秀梅等 |
| 9787302136798 | C 程序设计案例教程 | 王岳斌等 |
| 9787302115816 | C 语言程序设计教程 | 王敬华等 |
| 9787302112136 | Delphi 程序设计教程 | 杨长春等 |
| 9787302104001 | Excel 在数据管理与分析中的应用 | 杜茂康 |
| 9787302084945 | Internet 应用基础教程 | 徐详征等 |
| 9787302112211 | Internet 实用技术与网页制作 | 孙芳等 |
| 9787302120551 | Java 2 程序设计基础 | 陈国君等 |
| 9787302123224 | Java 程序设计之网络编程 | 李芝兴等 |
| 9787302127291 | PowerBuilder 数据库应用开发技术 | 卢守东 |
| 9787302134879 | Protel 电路设计教程(第 2 版) | 江思敏等 |
| 9787302140559 | SolidWorks 及 Cosmos/Motion 机械仿真设计 | 张晋西 |
| 9787302124344 | SPSS 统计分析实例精选 | 周爽 |
| 9787302105671 | Visual Basic 语言程序设计教程与实验 | 丁学钧等 |
| 9787302091349 | Visual Basic 程序设计与应用开发案例教程 | 曾强聪 |
| 9787302104322 | Visual Basic 程序设计综合教程 | 朱从旭等 |
| 9787302081449 | Visual C++程序设计——基础与实例分析 | 朱晴婷等 |
| 9787302123125 | Visual FoxPro 8.0 实用教程 | 李明等 |
| 9787302138389 | Visual FoxPro 数据库应用 | 田喜群等 |
| 9787302129967 | Visual FoxPro 程序设计 | 程学先等 |
| 9787302133216 | Visual FoxPro 程序设计基础 | 余坚 |
| 9787302133629 | Visual FoxPro 程序设计实验与学习指导 | 余坚 |
| 9787302132509 | Visual FoxPro 数据库基础教程 | 姜桂洪等 |
| 9787302098560 | Visual FoxPro 数据库应用教程与实验 | 徐辉等 |
| 9787302101185 | Web 技术导论 | 郝兴伟 |
| 9787302095453 | Windows 程序设计技术 | 刘腾红等 |
| 9787302088530 | 办公自动化概论 | 张锐昕等 |
| 9787302137511 | 操作系统教程与实验 | 胡明庆等 |
| 9787302102519 | 操作系统实验教程(Windows 版) | 姚卫新 |
| 9787302134626 | 程序设计基础(C 语言版) | 赵妮 |
| 9787302132325 | 大学计算机基础(含实验) | 王长友 |
| 9787302101802 | 单片机原理、接口及应用——嵌入式系统技术基础 | 李群芳等 |
| 9787302124542 | 电子档案管理基础 | 王萍等 |
| 9787302102526 | 多媒体技术毕业设计指导与案例分析 | 贺雪景等 |
| 9787302083771 | 多维数据分析原理与应用 | 姚家奕等 |
| 9787302101178 | 计算机辅助设计教程 | 张秉森等 |
| 9787302107910 | 计算机控制技术 | 姜学军 |
| 9787302100881 | 计算机外围设备 | 张钧良 |
| 9787302082057 | 计算机网络技术基础教程 | 刘四清等 |
| 9787302133025 | 计算机网络技术及应用 | 王中生等 |

| 书　　号 | 书　　名 | 作　　者 |
| --- | --- | --- |
| 9787302143338 | 计算机网络技术及应用教程 | 杨青等 |
| 9787302080732 | 计算机网络技术教程——基础理论与实践 | 胡伏湘等 |
| 9787302120193 | 计算机网络教程 | 王群 |
| 9787302140108 | 计算机网络实用技术教程 | 李冬等 |
| 9787302118619 | 计算机网络与通信 | 陈向阳等 |
| 9787302104926 | 计算机网络与应用 | 石良武 |
| 9787302110453 | 计算机维修技术 | 易建勋 |
| 9787302082392 | 计算机信息技术应用基础 | 杜茂康等 |
| 9787302109341 | 计算机信息技术应用教程 | 彭宗勤等 |
| 9787302112563 | 计算机应用基础 | 刘毅等 |
| 9787302132608 | 计算机应用技术基础 | 范慧琳等 |
| 9787302133155 | 计算机应用技术学习指导与实验教程——例题精解与练习、上机实践 | 范慧琳等 |
| 9787302090731 | 计算机英语实用教程 | 张强华 |
| 9787302119715 | 计算机硬件技术基础 | 曹岳辉等 |
| 9787302086307 | 计算机与网络应用基础教程 | 朱根宜 |
| 9787302091929 | 建筑 CAD 技术应用教程 | 吴涛 |
| 9787302087571 | 局域网技术与应用 | 李琳 |
| 9787302140696 | 局域网与城域网技术 | 王文鼐等 |
| 9787302089070 | 科技情报检索 | 田质兵等 |
| 9787302133735 | 面向对象程序设计与 Visual C++ 6.0 教程题解与实验指导 | 陈天华 |
| 9787302123118 | 面向对象程序设计与 Visual C++ 6.0 教程 | 陈天华 |
| 9787302090700 | 面向对象技术与 Visual C++ | 甘玲 |
| 9787302123231 | 面向对象技术与 Visual C++学习指导 | 甘玲等 |
| 9787302116981 | 软件技术基础教程 | 周肆清等 |
| 9787302133766 | 实用计算机技术——公安司法应用实践 | 汤艳君等 |
| 9787302142157 | 数据结构——C++语言描述 | 朱振元等 |
| 9787302140757 | 数据库及其应用 | 肖慎勇等 |
| 9787302104728 | 数据库及其应用学习与实验指导教程 | 肖慎勇等 |
| 9787302142966 | 数据库系统及应用(Visual FoxPro)第二版 | 邓洪涛 |
| 9787302086253 | 数据库系统及应用(Visual FoxPro) | 邓洪涛 |
| 9787302124962 | 数据库与网络技术 | 翟延富 |
| 9787302128649 | 数据通信与网络应用 | 吴金龙等 |
| 9787302091295 | 统计分析方法——SAS 实例精选 | 周爽 |
| 9787302124795 | 图形图像处理应用教程 | 张思民等 |
| 9787302143086 | 网络工程规划与设计 | 陈向阳等 |
| 9787302124300 | 网络基础与应用实务教程 | 段宁华 |
| 9787302142690 | 网络医学信息应用 | 刘汉义等 |
| 9787302115595 | 网络远程教学技术基础(含上机指导) | 黄景碧等 |
| 9787302091875 | 网页设计教程 | 侯文彬等 |
| 9787302101819 | 网站建设——基于 Windows Server 2003 和 Linux 9 | 葛秀慧 |
| 9787302103417 | 微机组装与维护 | 查志琴等 |
| 9787302120513 | 信息检索 | 陈雅芝 |
| 9787302093619 | 运筹学算法与编程实践——Delphi 实现 | 刘建永等 |
| 9787302112006 | 中文信息处理技术——原理与应用 | 李宝安等 |